EU COMMUNITY

ICT-2013.5.4 ICT for Governance and Policy Modelling

EU COMMUNITY MERGES ICT AND SOCIAL MEDIA NETWORKING WITH ESTABLISHED ONLINE MEDIA AND STAKEHOLDER GROUPS TO CULTIVATE TRANSPARENCY, ENHANCE EFFICIENCY AND STIMULATE FRESH IDEAS FOR EU POLICY-MAKING

Deliverable D4.3.2

Policy Component Prototype (Second Version)

Editor(s):	Miltiadis Kokkonidis, Aggeliki Androutsopoulou, Yannis Charalambidis
Responsible Partner:	INTRASOFT International SA
Status-Version:	Final – v1.0
Date:	14/09/16
EC Distribution:	R

Project Number:	611964
Project Title:	EU COMMUNITY

Title of Deliverable:	Policy Component Prototype (Second Version)
Date of Delivery to the EC:	14/09/16

Workpackage responsible for the Deliverable:	WP4 – Policy Modelling and Impact Assessment Component
Editor(s):	Aggeliki Androutsopoulou, Yannis Charalabidis, Miltiadis Kokkonidis
Contributor(s):	INTRA, AEGEAN
Reviewer(s):	AEGEAN, EURACTIV
Approved by:	All Partners

Abstract:	The document describes the second version of the Policy Component Prototype developed as part of the EU Community platform. It outlines the design of the two subcomponents, the Ontology Module and the Predictions Module.
Keyword List:	Policy modelling, impact assessment, system dynamics, blended expert-machine approach, ontology, design, specifications, simulation

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Modification Reason	*Modified by*
V0.1	04/03/2016	Initial version, based on D4.3.1	INTRA
V0.2	11/03/2016	Updates to Chapters 3 and 4 based on developments in D4.4.1	INTRA
V0.3	15/03/2016	Updates to Chapters 1 and 2 based on developments in D4.4.1	INTRA
V0.4	17/03/2016	Updates based on work in the second prototype	INTRA
V0.5	21/07/2016	Addition of section about policy process ontologies and the export agent. Further updates based on work in the second prototype	INTRA, AEGEAN
V0.6	19/08/2016	Restructuring of deliverable so as to focus on the expansion of score of the second version of the Policy Component. Those changes were meant to make it easier to complete D4.4.2 as a continuation of D4.4.1 and D4.3.2 as a continuation of D4.3.1. Updates on practically all sections.	INTRA
V0.7	25/08/2016	Minor edits on the Simulation Subsystem section.	AEGEAN
V0.8	26/08/2016	Draft prepared for internal review. Minor edits throughout the deliverable. Finalisation of Conclusions chapter.	INTRA, AEGEAN
V0.9	12/09/2016	Revision based on comments by partners and internal reviewers	INTRA
V1.0	14/09/2016	Final version sent to the EC	INTRA

Contents

EXECUTIVE SUMMARY 9

1 INTRODUCTION 11

1.1 OBJECTIVES AND PURPOSE 11

1.2 RELATION TO OTHER WORK PACKAGES/DELIVERABLES 11

1.3 STRUCTURE OF THE DOCUMENT 12

2 THE POLICY COMPONENT: ARCHITECTURE AND IMPLEMENTATION TECHNOLOGIES 13

2.1 OVERVIEW 13

2.2 POLICY COMPONENT ARCHITECTURE 14

2.3 IMPLEMENTATION TECHNOLOGIES 15

3 ONTOLOGY MODULE 17

3.1 SCOPE 17

3.2 USER ROLES 17

3.3 SECTIONS AND TOPICS 18

3.3.1 EDITING SECTIONS AND TOPICS 19

3.3.2 SECTIONS AND TOPICS IN THE EU COMMUNITY PROJECT 22

3.4 CONCEPTS 22

3.4.1 EDITING CONCEPTS HIERARCHIES 23

3.4.2 CONCEPT HIERARCHIES IN THE EU COMMUNITY PROJECT 24

3.5 CONCEPTS-TOPIC LINKING 25

3.5.1 LINKING CONCEPTS WITH TOPICS 25

3.5.2 CONCEPT-SECTION AND CONCEPT-TOPIC LINKING IN EU COMMUNITY 26

3.6 POLICY PROCESSES 27

4 PREDICTIONS MODULE 30

4.1 SCOPE 30

4.2 USER ROLES 30

4.3 POLICY PROCESSES AND LEGISLATIVE PROCEDURE MODELLING IN THE PREDICTIONS MODULE 30

4.4 HYBRID PREDICTIONS SUBSYSTEM 31

4.4.1 WHY A HYBRID PREDICTIONS SUBSYSTEM? 31

4.4.2 INTRODUCING PREDICTION GAMES 32

4.4.3 THE FINAL OUTCOME PREDICTION GAME 33

4.4.4 THE END DATE PREDICTION GAME 36

4.5 SIMULATION SUBSYSTEM 37
4.5.1 MODEL DEFINITION 37
4.5.1.1 CAUSAL LOOP DIAGRAM 38
4.5.1.2 STOCK AND FLOW DIAGRAM 40
4.5.2 IMPLEMENTATION AND INTEGRATION 41
5 CONCLUSIONS – FUTURE WORK 43
APPENDICES 45
APPENDIX A: INTEGRATION WEB APIS 45
A.1 POLICY DOMAIN ONTOLOGY WEB API 45
A.1.1 PROTOCOL 45
A.1.2 SECTIONS AND TOPICS ACTION MESSAGES 46
A.1.3 CONCEPT-RELATED ACTION MESSAGES 47
A.1.4 CONCEPT-TOPIC LINKING-RELATED ACTION MESSAGES 49
A.2 HYBRID PREDICTIONS SUBSYSTEM WEB API 49
A.3 SIMULATION SUBSYSTEM WEB API 50
APPENDIX B: POLICY PROCESS ONTOLOGY 52
B.1 POLICY DOMAIN ONTOLOGY 52

List of Figures

FIGURE 1: WP4 TASKS AND DELIVERABLES 11
FIGURE 2: WPS, TASKS AND THEIR DEPENDENCIES 12
FIGURE 3: THE POLICY COMPONENT MODULES 14
FIGURE 4: POLICY COMPONENT ARCHITECTURE 14
FIGURE 5: SECTIONS & TOPICS HIERARCHY EDITOR 19
FIGURE 6: CONCEPT HIERARCHY EDITOR 24
FIGURE 7: CONCEPTS-TOPICS LINKING MATRIX 26
FIGURE 8: AN EXAMPLE VISUALISATION OF A POLICY PROCESS 28
FIGURE 9: DOCUMENT GENERATION LOOP 39
FIGURE 10: POSITIVE AND NEGATIVE DOCUMENTS VARIATION 39
FIGURE 11: THE CAUSAL LOOP DIAGRAM OF THE SIMULATION MODEL 40
FIGURE 12: STOCKS AND FLOWS DIAGRAM 41

List of Tables

TABLE 1: DEFINITIONS, ACRONYMS AND ABBREVIATIONS 8
TABLE 2: THE POLICY PROCESS METRICS 38

Definitions, Acronyms and Abbreviations

Table 1: Definitions, Acronyms and Abbreviations

Acronym	Title
AJAX	Asynchronous JavaScript + XML
AJAJ	Asynchronous JavaScript and JSON
API	Application Programming Interface
JSON	JavaScript Object Notation
OWL	Web Ontology Language
RDF	Resource Description Framework
SD	System Dynamics
SKOS	Simple Knowledge Organisation System
UI	User Interface
XML	Extensible Markup Language

Executive Summary

The Policy Component embodies the contribution of WP4 to the EU Community project. This contribution includes bringing into the project or developing relevant know-how in the area of ontologies, semantic web technologies, policy process and legislative procedure modelling, simulation modelling and system dynamics in particular, and a hybrid approach to policy process predictions combining past data with expert input.

The Policy Component provides two largely independent sub-components with distinct functionality and purpose: the Ontology Module and the Predictions Module.

The Ontology Module is concerned with the maintenance of:

- The EU Community Sections and Topics Hierarchy: Sections correspond to broad EU policy domains. For example, in a commercial deployment of a service based on the EU Community Platform or a derivative thereof, there could be a Section for each DG. For the purposes of the Project Pilots (WP7), three Sections have been chosen: Energy Union, Innovation and Entrepreneurship, Future of EU. A Section is broken down into a number of Topics. Topics are meant to correspond to more specific areas of EU policy making. For example, the Energy Union Section can be broken down into topics such as Renewable Energy, Energy Efficiency, Energy Supply Security, etc. The Sections and Topics hierarchy provides a very basic and fundamental organisation of information in the EU Community Platform. Any changes to it are expected to be the result of serious thought and strategic planning; changes are expected to be infrequent and to be carried out by a very small group of users with special administrative privileges.

- The EU Policy Concept Ontologies (Concepts and their relations): Whereas the EU Community Platform has a simple 2-level hierarchy of Sections and Topics for internal purposes, it gathers documents from a number of sources documents which may include (textual content) or be indexed (indexing meta-data) on the basis of one or more terms or concepts. A number of text processing tasks can make use of such terms or concepts if EU Community is aware of them and how they are related to one another. A term is understood to be a mere textual representation, whereas a concept is an abstract notion that is related to one or more textual representations in one or more languages. Matching on the level of concepts is therefore much more useful in the context of a project with a view to being expanded in the future to support EU's multi-linguality. In addition to defining the relation of concepts with their representations (currently only in English), the Ontology Module supports maintaining relations between concepts.

- Topic-Concept Linking: Whereas the EU Community Platform has a simple 2-level hierarchy of Sections and Topics for internal purposes, it gathers documents from a number of sources documents which may be indexed on the basis of different terms or concepts. For instance, EurLex indexes documents and legislative procedures on the basis of concepts defined in EuroVoc, a multilingual collection of concepts with relations such as BT(Broader Than), RT(Related To) etc. defined between them. They key

ideas are that (1) the Ontology Module allows the EU Community Platform to defines the Sections and Topics it will use (2) various document sources may be using different terms and concept collections to index their documents (3) neither the EU Community Platform nor the sources it uses have their decisions about Sections & Topics (EU Community Platform) or indexing terms/concepts collections (document sources) depending on one another i.e. their designs are perfectly decoupled (4) the Ontology Module allows Concepts to be linked with Sections and Topics, thus providing a way to categorise indexed documents from external sources (i.e. assign one or more Sections and Topics to them).

At the same time, in its second version, the Ontology Module exposes in semantic web standard formats, the Policy Process data maintained by the community using PolicyLine. This allows such work conducted within the framework of the EU Community project, and/or any possible research or commercial continuation, to be made publicly available allowing it to be reused in different projects.

The Predictions Module is concerned with predictions:

- The Hybrid Predictions Subsystem is concerned with predictions to questions such as "When will legislative procedure X be completed?" and "What will its outcome be?". It uses both data about past legislative procedures and any user input available (user predictions on the same questions) in order to provide answers.

- The Simulation Subsystem is concerned with predicting the dynamics of a policy discussion captured in the form of time-series data showing the predicted evolution of metrics such as Sentiment, Controversy, Awareness, Engagement etc.

1 Introduction

1.1 Objectives and Purpose

The objective of the current deliverable is to describe the second version of the Policy Component Prototype. For this purpose, it provides a breakdown of the Policy Component into (largely) independent modules with discrete functionality, purpose, and interfaces, and explores their common overarching architecture and integration paths with the rest of the EU Community platform.

1.2 Relation to other Work Packages/Deliverables

The current Deliverable D4.3.2 is the third deliverable of T4.2 ("Policy Component Design and Development"). In the first deliverable of T4.2, D4.2, the specifications for the Policy Component were laid out; D4.3.2 and its predecessor D4.3.1 describe the progress towards meeting those requirements at two distinct points in time, one near the middle (D3.2.1) and one near the end (D3.2.2) of the EU Community project.

WP4 has a separate testing and adaptations task, T4.3. The first deliverable of T4.3, D4.4.1 ("Test cases, Adaptations and Evaluation Results (first version)") followed D4.3.1 and presented both an evaluation of its achievements as well as a description of advances in certain algorithms which resulted in significantly better results, especially with respect to the Statistical Predictor. A follow-up deliverable of T4.2, D4.4.2, will follow the present deliverable and will have the same role with respect to it, as D4.4.1 had with respect to D4.3.1.

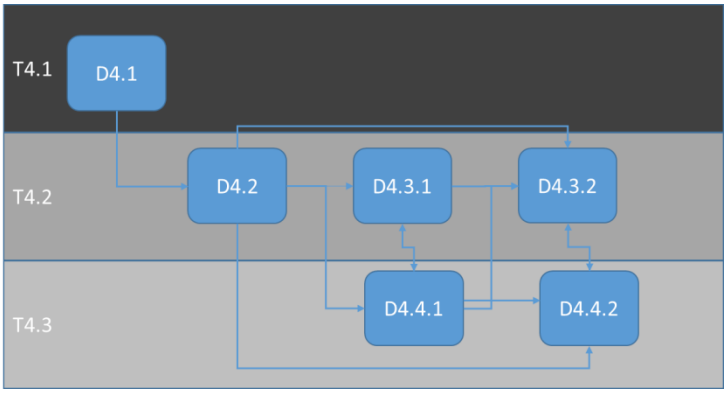


Figure 1: WP4 Tasks and Deliverables

With respect to the broader, project-level, picture, the requirements reported in D2.4 ("Community requirements and specifications") have been taken into consideration, as well as D6.1 ("Platform Architecture Design").

At the same time, WP4 has played a role on both formulating the Consortium's current views across the board on matters pertaining to its subject matter, on relevant requirements, on the platform's architecture, the common database and the mode of interoperability. In other words, there has been continuing dialogue

between WP4 and WP2, WP3, WP5 and WP6. The outcome of this ongoing dialogue is reflected in practically all year 2 and year 3 deliverables of the project.

Finally, it is worth mentioning the significance of the project review meetings (Year 1, Year 2, Interim Year 3) have had on shaping the developments in WP4, as evident both in D4.31 & D4.4.1, and in D4.3.2 (the present deliverable) and D4.4.2 (the following and final deliverable of the WP).

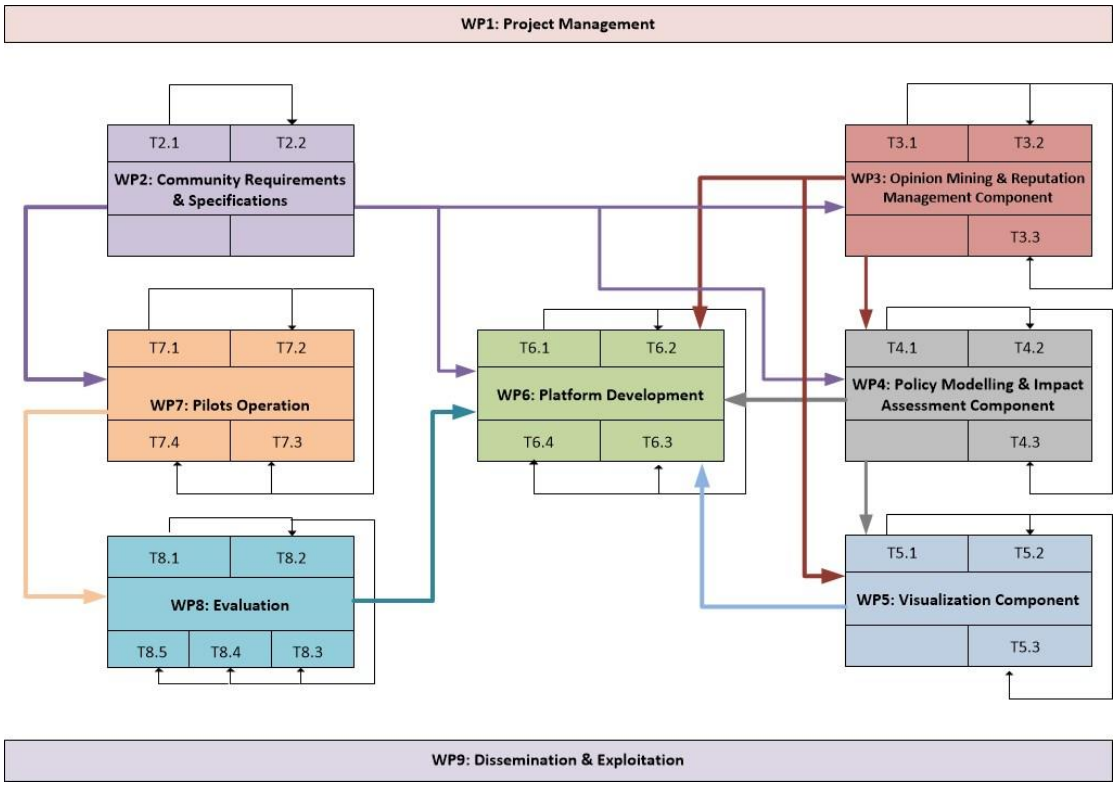


Figure 2: WPs, Tasks and their Dependencies

1.3 Structure of the Document

The deliverable is structured as follows:

Chapter 2 presents an overview of the current version of the Policy Component Prototype, its architecture and implementation technologies.

Chapter 3 examines more closely the current version of the Ontology Module, consisting of the Policy Domain Ontology Builder and the Policy Domain Ontology Server, present also in its first version, as well as the newly introduced Policy Process Semantic Web Publication Agent.

Chapter 4 examines more closely the Predictions Module and its two distinct subsystems: the Hybrid Predictions Subsystem and the Simulation Subsystem.

Chapter 5 summarises conclusions on the current development stage, progress achieved and future steps considered.

2 The Policy Component: Architecture and Implementation Technologies

2.1 Overview

Two largely independent sub-components with distinct functionality and purpose make up the Policy Component:

- the Ontology Module (which includes the Policy Domain Ontology Builder and the Policy Domain Ontology Server, as well as a subsystem not present in the first prototype, namely the Policy Process Semantic Web Publication Agent) and
- the Predictions Module (consisting of the Hybrid Predictions Subsystem and the Simulation Subsystem).

The two modules are presented in more detail below.

The Ontology Module consists of:

- The Policy Domain Ontology Builder, a specialised tool which allows a small group of designated domain experts (Ontology Builders) to provide a conceptual map of policy topics. The ontology is a valuable resource that may be used in a number of text-related tasks. Additionally, the Policy Domain Ontology Builder allows the Ontology Administrator(s) to edit the Sections and Topics hierarchy, which is the basis of the organisation of both the ontology itself, but also of the content in the EU Community Platform (in PolicyLine: policy processes and their documents belongs to a Topic and a Topic to a Section, in EurActory: sections and topics are the areas of expertise the RMS ranks users and non-users on).
- The Policy Domain Ontology Server, an interoperable ontology management system tailored to the needs of the Policy Domain Ontology Builder and the EU Community project.
- The Policy Process Semantic Web Publication Agent, a web-based server that makes available the EU Community policy processes, which PolicyLine maintains and visualises, as semantic web open data.

The Predictions Module provides:

- A Hybrid Predictions Subsystem, which utilises a combination of machine understanding of legislative procedures, historical data and expert estimates, if available, in order to produce predictions about the unfolding of legislative procedures (e.g. "When will legislative procedure X be completed?" and "What will its outcome be?").
- The Simulation Subsystem, which utilises System Dynamics predictive modelling to produce predictions of the dynamics of a policy discussion captured in the form of time-series data showing the predicted evolution of metrics such as Sentiment, Controversy, Awareness, Engagement etc. The visualisation of such data can be used to answer questions such as: "Will interest wane in a few weeks?", "Will a future-planned event boost awareness?" etc.

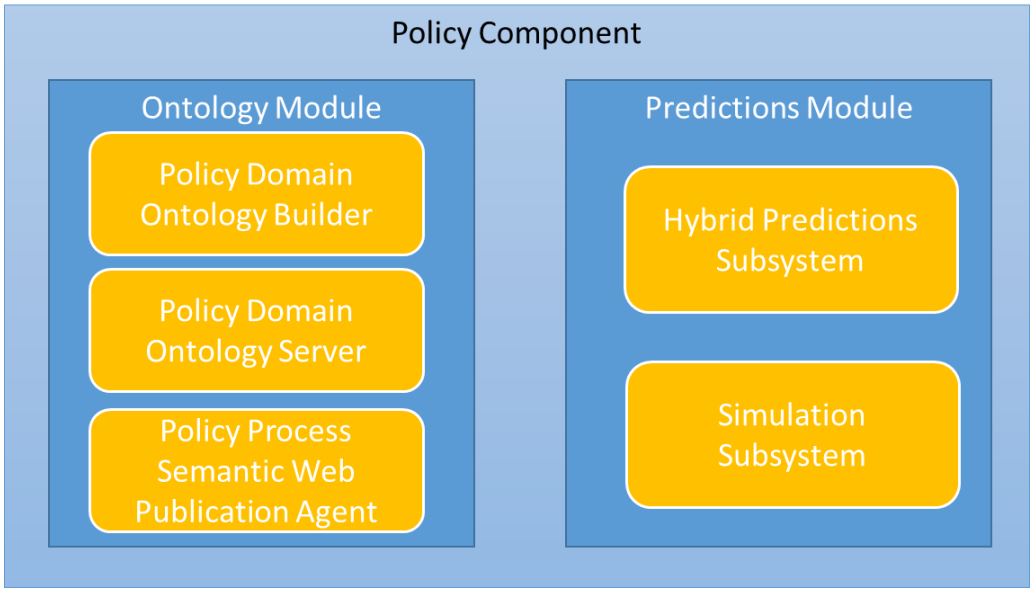


Figure 3: The Policy Component Modules

Details regarding each of the two modules are given in the following chapters.

2.2 Policy Component Architecture

The Policy Component is implemented as a single ASP.NET C# MVC Web Application. This web application has four parts: the Policy Domain Ontology Builder, the Policy Domain Ontology Server, the Hybrid Predictions Subsystem, and the Simulation Subsystem. The latter three expose Web APIs that can be used by PolicyLine and/or even independent third-party components not developed as part of the EU Community project.

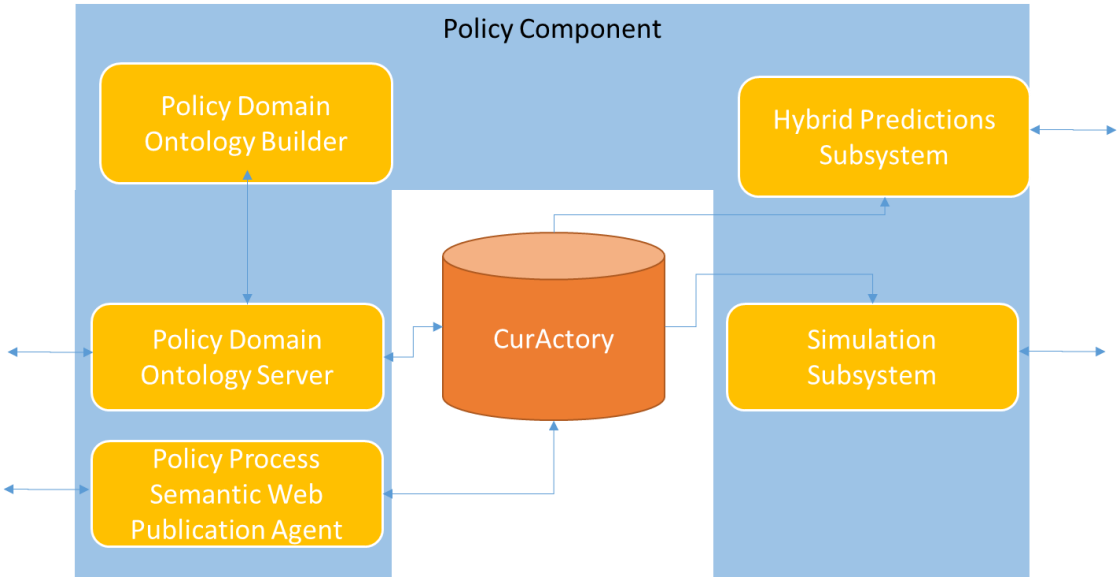


Figure 4: Policy Component Architecture

A brief description of the five parts of the Policy Module is given below:

- The Policy Process Semantic Web Publication Agent is a web application that exposes each policy process in CurActory as semantic web open data in accordance with the EU Community Policy Process Ontology.
- The Policy Domain Ontology Builder is an AJAX (more accurately, AJAJ)/Javascript/HTML application that can be used for creating and modifying Sections, Topics, Concepts and links between them.
- The Policy Domain Ontology Server implements the business logic of the Policy Component with regard to operations on Sections, Topics, Concepts and links between them; it supports the Policy Domain Ontology Builder's operation and provides data persistence via CurActory (with which it communicates directly rather that via the CurActory Web API). The Policy Domain Ontology Server exposes its functionality by means of a Web API which serves the Policy Domain Ontology Builder but may also support other future interoperability needs.
- The Hybrid Predictions Subsystem exposes a simple Web API that provides answers to specific questions regarding the progress of ongoing legislative procedures. To do this, it uses a combination of historical data stored in CurActory by the EURLex Legislative Procedure Crawler and expert opinions stored in CurActory by PolicyLine. The Hybrid Predictions Subsystem's Web API was designed to be accessed by PolicyLine, but will also be available for use by third-party components.
- The Simulation Subsystem exposes a simple Web API that provides timeseries predictions on the development of aspects of a discussion. It works by performing a simulation of the discussion's development based on a Systems Dynamics model for which the parameters defining the initial state of the model are obtained from CurActory. The Web API of the Simulation Subsystem was also designed with the primary purpose of being accessed by PolicyLine, but, like the Hybrid Predictions Subsystem Web API, will also be available for use by third-party components.

Note: Discussions between Consortium members lead to a change in the mode of interoperability between the EU Community platform's components. This change affected the Policy Component's architecture. In summary, the main difference with the architecture envisaged in D6.2 is that the Policy Component now has direct access to CurActory, i.e. the EU Community project's database, eliminating the need for communication via an intermediate CurActory Web API updated on demand by the integration workpackage (WP6) leader. This has helped speed up collaboration between WP5 and WP4.

2.3 Implementation Technologies

The primary implementation technology of the Policy Component is ASP.NET MVC 5.2. The framework natively supports both JSON and XML serialisation. For any non-automatic JSON parsing/serialisation Newtonsoft's JSON.Net 6.0 library is used. For access to CurActory, Microsoft's Entity Framework 6.1 is used as an object-relational mapping technology providing type safety and the ability to use LINQ to express database queries. For serialisation/parsing of RDF data, the dotnetRDF v1.0.8 library is used. C# is the primary implementation language.

However, the System Dynamics model is in Vensim .mdl format and the code executing it is in Python 2.7 (using a pysd v2.1, a Systems Dynamics library for Python). Also, Javascript is the programming language used in the client-side Policy Domain Ontology Builder. AngularJS v. 1.2.28, jQuery v.1.11.1 and Bootstrap v.3.3.31 are the main Javascript libraries used. The current presentation layer web standards, HTML 5 and CSS 3, are also used. Overall, a broad range of technologies and libraries has been used in the Policy Component prototype; these technologies and libraries have worked very well together each serving their specific chosen role in the implementation more than adequately.

3 Ontology Module

3.1 Scope

The Ontology Module comprises of the Policy Process Semantic Web Publication Agent, the Policy Domain Ontology Builder and the Policy Domain Ontology Server.

- The Policy Domain Ontology Builder, is a specialised tool meant to be used by a select group of topic experts (moderators) to:
    1. curate the simple two-level hierarchy of Sections (areas of EU policy) and Topics (specific topics under the broader areas of EU policy represented by Sections)[1]
    2. curate a conceptual map (of arbitrary size, hierarchical complexity and informational detail) for each area (Section) of EU policy and its topics.
- The Policy Domain Ontology Server implements the business and supplies the interfaces necessary for the Policy Domain Ontology Builder to perform its tasks.
- The Policy Process Semantic Web Publication Agent exposes each policy process in CurActory as semantic web open data. It does so in accordance with the EU Community Policy Process Ontology which captures the project's modelling of policy processes.

3.2 User Roles

The Policy Process Semantic Web Publication Agent does not involve registered users and roles, as its purpose is to expose policy process data as open data for which no access control is required.

There are two user roles of interest to the Policy Domain Ontology Builder's (and the Policy Domain Ontology Server's) access control system:

1. The role of **ontology builders**: this role carries the privileges necessary for creating, updating, approving, deleting and undeleting Concepts, as well as linking them to Sections and Topics. It also allows limited access to the Sections and Topics hierarchy.

---

[1] The current intention is that after the completion of the current project, the EU Platform will be used with Sections in direct correspondence to DGs and their policy areas. However, for the duration of the project there will be only three Sections, each corresponding to a Pilot Topic (see WP7). The tool herein described is dedicated to defining the organization of content in the platform and as such will be useful also in the required transition, not only in the re-definition of the Sections & Topics hierarchy, but also in the much more challenging task of enriching the EU Platform's ontology (the set of concepts it's AI "understands").

2. The role of **ontology administrators:** in addition to the above privileges, ontology administrators may also create, update, approve, delete and undelete Sections and Topics.

Ordinary users and non-users do not have access to the tool (as mentioned earlier it is a specialised tool with a very limited user base). Rather, they interact with Sections, Topics and Concepts through the main tools of the EU Community platform: EurActory and PolicyLine.

3.3 Sections and Topics

The Sections and Topics hierarchy is a simple two-level hierarchy with Sections at the upper level and Topics at the lower one. Topics are meant to correspond to specific areas of EU policy making, whereas Sections are meant to correspond to broader areas of EU policy making. In other words, Sections group together a number of Topics under a common heading. For instance, the Section "Energy Union" has a number of topics such as "Internal Energy Market", "Energy Efficiency", "Research and Innovation in Energy" under it (Policy Processes come under Topics and Documents under Policy Processes, but maintenance of the platform's content at that level is left to PolicyLine ordinary users and moderators.)

Each Section and each Topic can be in one of the following states: Proposed, Verified, Deleted.

The Ontology Module has to cater for the possibility that ordinary users may propose new Topics (and possibly even Sections) via user-facing tools within the core or an extended (via third party tools) EU Community Platform. The idea is that when performing actions normally involving selecting existing Sections and Topics an option is given to ordinary users to propose new Topics (and possibly even Sections). Proposed Sections and Topics enter the topic/sub-topic hierarchy in the Proposed state, which means they are not visible to any component or module other than the Policy Domain Ontology Builder.

Only an **ontology administrator** may verify a proposed Topic (or Section), thus changing its state to Verified, at which point it officially becomes part of the Sections and Topics hierarchy and is visible throughout the EU Community platform. This is done using the Policy Domain Ontology Builder.

By default, any Section or Topic added by a user with **ontology administrator** privileges automatically enters the Verified state and therefore needs no additional verification step. However, if **ontology administrators** so wish, they may mark a new Section or Topic they add as being in the Proposed state, so that either a colleague or themselves at a later stage may consider either verifying it or removing it altogether. Users only having the **ontology builder** role may also add Sections and Topics, but any Sections and Topics they add will initially be in the Proposed state; such users may not perform any other kind of action that modifies the Sections and Topics hierarchy, as their main responsibility is the Concepts ontology.

Any Section and Topic may be marked as deleted and thus be made invisible to any component or module other than the Policy Domain Ontology Builder. If the deletion marks is removed, the Section or Topic returns to its prior state (Verified or Proposed).

3.3.1 Editing Sections and Topics

An **ontology administrator**, logged in the Policy Domain Ontology Builder, may begin to curate the Sections and Topics hierarchy by selecting the "Sections and Topics" menu option. This option is also available to **ontology builders**, however, as already mentioned, the only editing functionality available to them is the option to add new topics in the Proposed state (i.e. propose new topics).

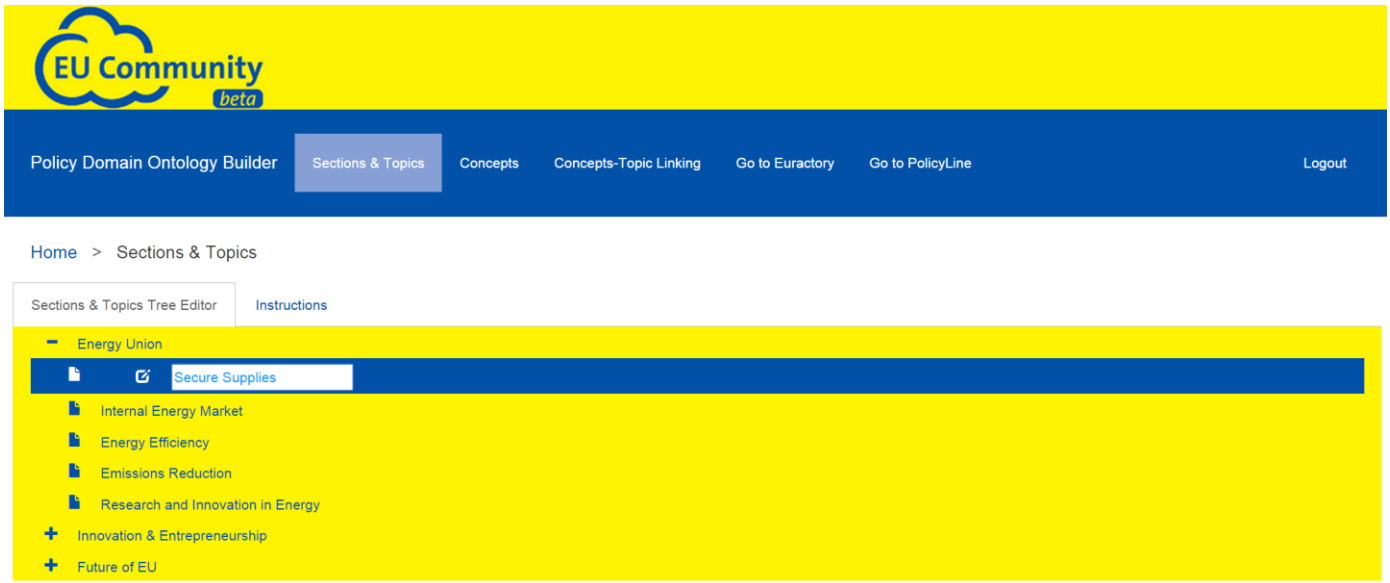


Figure 5: Sections & Topics Hierarchy Editor

There are two tabs in the Sections & Topics page: one containing an editor control facilitating the curation of the Sections & Topics hierarchy and another containing instructions. A similar design is used throughout the Policy Domain Ontology Builder.

As per D6.2 requirements, the hierarchy editor has been designed with certain design goals in mind:

1. To be easy to use
2. To allow very fast data entry

The latter point is of outmost importance when editing Concept hierarchies, which tend to be significantly larger and more complex (more hierarchical levels and more relations). Nevertheless, we will first discuss the way the user interacts with the hierarchy editor here, in the context of the Sections and Topics hierarchy.

The distinguishing feature of the way the Policy Domain Ontology Builder handles the task of hierarchies editing is that it introduces a very simple markup language based on a *small* selection of special symbols. There are a number of examples of successful applications of this concept.

- Twitter very successfully introduced special markup on the basis of the "@", "#", and "$" symbols:

  - @username constitutes a *mention* of that user in a Tweet (@reply tweets being a special case of mention; see the relevant Twitter documentation[2])
  - # in front of a sequence of letters and a few other permitted characters (typically word or a phrase without spaces) constitutes a *hashtag* (examples: #love, #EU, #StopACTA), a very simple and efficient way of allowing users to index their tweets so that they will be included in searches for any of the hashtags they include
  - $ followed by a company ticker symbol constitutes a *cashtag* (examples: $APPL, $MSFT, $TWTR), a lesser known feature of Twitter, similar to hashtags but for company mentions used, for example, in relation to financial news that may be of interest to the stock market but also in other contexts

- A number of messaging applications (including email clients, Skype and many others) interpret the symbols "*" and "_" in a special manner:
  - "*" before and after a piece of text is interpreted as an indication that it should be rendered in **bold** ("*bold*") whereas
  - "_" before and after a piece of text is interpreted as an indication that it should be rendered in *italics* ("_italics_").

  Moreover, they interpret certain symbol sequences as emoticons and render them accordingly (example: ":-)" is rendered as a smiling face "☺"). Though, these conventions can be traced to technological shortcomings in the early days of computing in which these conventions originate in (the use of GUI terminals was a rarity and communication was by means of plain text), the fact is that their simplicity and convenience has lead to their adoption in modern software, often without an alternative interface to provide the same functionality.

- A small ad hoc markup conventions are also used by individuals or collaborating groups in editing text documents, spreadsheets or code usually to indicate that a piece of text or information or code is work in progress that needs further attention before being published, sent to a third party, or, in the case of code, being put to production.

An important common feature of all these conventions, whether supported by the software platform on which they are used or not, is that they blend in with the task of editing text, rather than interrupt it with a selection from some menu using, for example, the mouse.

The other feature they share is that where there are such conventions, there are *very few*[3]. This is important, because the usability and human-computer

[2] Twitter Inc., "What are @replies and mentions?" URL: https://support.twitter.com/groups/52-notifications/topics/212-replies-mentions/articles/14023-what-are-replies-and-mentions# Last Retrieved: 01/05/2015

interaction efficiency benefits of not having to change the mode of input (keyboard-mouse-keyboard again) entirely lost if users have to think about the conventions or look them up.

In editing the Sections and Topics hierarchy, the expert users with the appropriate assigned role, **ontology administrator**, will have to remember the following simple conventions:

- If the Section or Topic starts with a minus sign "-" it is in the DELETED state, and therefore not visible in any other EU Community Platform component.
- If the Section or Topic starts with a minus sign "*" it is in the PROPOSED state, and therefore not visible in any other component of the EU Community Platform (except perhaps in the context where it was proposed with appropriate indication of its status).
- If neither of the above is the case, the Section or Topic is in the APPROVED state and is used throughout the platform.

Ordinary users and moderators do not need to know these conventions as:

1. they are not responsible for deleting/undeleting topics
2. when ordinary users propose a topic in tools within the core or extended (inclusive of third-party components) EU Community Platform tools they just enter its name.

Ontology builders merely need to be aware of the asterisk convention in order to recognise it when they see it; when they add a topic in the Policy Domain Ontology Builder it is automatically treated as a proposed topic (the asterisk is prepended automatically).

The data entry interface is particularly simple and does not stray far from, say, editing a list of bullets and sub-bullets in MS Word, a task that can be done quickly using no other input device than the keyboard.

- To navigate around the hierarchy the **ontology administrator** may use either the up and down arrow keys, or point and click on the tree.
- Once an item is selected (a Section or a Topic), a textbox appears allowing the user to immediately modify its text.
- To mark an item as deleted, the user inserts a minus as its first character. To remove the deletion mark, the user simply removes the preceding minus sign.
- To remove an item altogether from the hierarchy, the **ontology administrator** simply deletes its textual content. This is only allowed for topics that have been entered in the hierarchy but not used in any context (except the Section/Topic proposal contexts in EurActory and PolicyLIne).

---

[3] Emoticon sequences are a notable exception but this can be explained on the basis of both their more playful purpose and context of use and their distinguishing property that they can be, with some practice perhaps, interpreted on the basis of their appearance.

Topics and Sections that have been assigned as fields of expertise or linked to documents or policy processes cannot be removed from the hierarchy in this manner. They can only be marked as deleted, as described above, so that they cannot be used again, but the old links with other entities in the EU Community Platform will remain.

- To approve a Section or Topic that is in the PROPOSED state and starts with an asterisk ("*"), the **ontology administrator** simply removes that asterisk. Likewise, if he/she puts an asterisk at the front of the definition of a Section or Topic, this will be put in the PROPOSED state and will not be visible to any other tool in the platform.

- To create a new item, the **ontology administrator** or **ontology builder** presses Enter or Tab. This results in a pop-up window appearing where the text of the new item may be entered. If the user not an **ontology administrator**, the item will be put in the PROPOSED state.

3.3.2 Sections and Topics in the EU Community Project

The Sections and Topics hierarchy serves two key purposes:

1. It serves to organise content in the EU Community platform and more specifically in PolicyLine (WP5). Note: There are three main levels of content organisation in PolicyLine: Sections, Topics, Policy Processes. WP4 is responsible for defining all three at a technical and conceptual level, but the creation and curation of Policy Processes is in the purview of PolicyLine, whereas the creation and curation of Sections and Topics is in the exclusive purview of the Policy Domain Ontology Builder.

2. Sections and Topics are also relevant to EurActory and the Reputation Management System as they serve as the areas of expertise for individuals listed there.

The Sections and Topics hierarchy provides a very basic and fundamental organisation of information in the EU Community Platform. Any changes to it are expected to be the result of serious thought and strategic planning. In summary, changes are infrequent but affect the entire platform. For the purposes of WP7 (Pilots), three Sections (with their corresponding Topics) will be defined, one for each pilot, but, as noted earlier, a richer Sections & Topics hierarchy, covering all aspects of EU policy, would be used under normal operation.

3.4 Concepts

The main function of Policy Domain Ontology Builder is to provide conceptual maps for each Section of EU policy and its Topics. These conceptual maps are hierarchies of Concepts each with one or more textual representations. These hierarchies, unlike the Sections & Topics hierarchy are of arbitrary complexity. For instance, the Concept "energy source" may have a child Concept "renewable energy source" which may in turn have a child Concept "biomass" which may itself have a child Concept "lignocellulosic biomass" which may itself have a number of child Concepts, each child Concept being more specific and narrow than its parent. The hierarchies are formed on the basis of the "More Generic Than / Broader Than" relation.

3.4.1 Editing Concepts Hierarchies

It is conceptually correct to think that EU Community supports a single concept hierarchy, but for the purposes of keeping its maintenance manageable, the task of maintaining it is presented to users as a task of maintaining one concept hierarchy per Section. A user with the **ontology builder** role (or the **ontology administrator** role which subsumes it), logged in the Policy Domain Ontology Builder, may curate any of these by selecting "Concepts" from the top options menu and then choosing the tab corresponding to a Section; the Section's concept hierarchy will appear and the user will be able to begin any related ontology curation tasks.

The editor used for Concept hierarchy editing is the same as the one used for editing the Sections & Topics hierarchy, minor parameterisation differences aside. However, whereas for the Sections & Topics hierarchy fast data entry is a desirable requirement, for concept ontology building it is an essential one. We have identified this particular challenge from a very early point in the project and have provided a user interface design that meets it, without sacrificing ease of use; this will be discussed further in D4.4.2. It is not the case that someone can begin entering ontologies without prior instructions, but this is because the purpose and rules for choosing the concepts and organising the ontology need to be understood first; the ontology editing tool itself is quite straightforward.

The user interface is similar, but not entirely identical, to that of the Sections & Topics hierarchy editor.

- To navigate around the hierarchy, the user may use either the up and down arrow keys, or point and click on the tree.
- Once an item is selected, a textbox appears allowing the user to immediately modify its text.
- A Concept may have multiple alternative textual representations; these are to be listed with a vertical bar "|" character in between them i.e. as a separator character. The preferred textual representation must be given first in the list. For example, the text "renewable energy sources | RES | renewables" provides three alternative textual representations for the same concept, with "renewable energy sources" being the preferred one.
- To remove a Concept from a Section's Concept hierarchy, the user simply deletes its textual content.
- To mark a Concept as deleted, the user inserts a minus as its first character. To remove the deletion mark, the user simply removes the preceding minus sign. A Concept that is marked as deleted, appears only in the Policy Domain Ontology Builder's ontology editor (so that it may be undeleted), but is otherwise ignored by the EU Community platform. Deletion affects the Concept itself rather than its inclusion in a particular Section's hierarchy.

- To create a new item, the user presses Enter or Tab. This results in a pop-up window appearing where the text of the new item may be entered. The pop-up offers the option of adding the new concept as a parent or as a sibling of the current one.

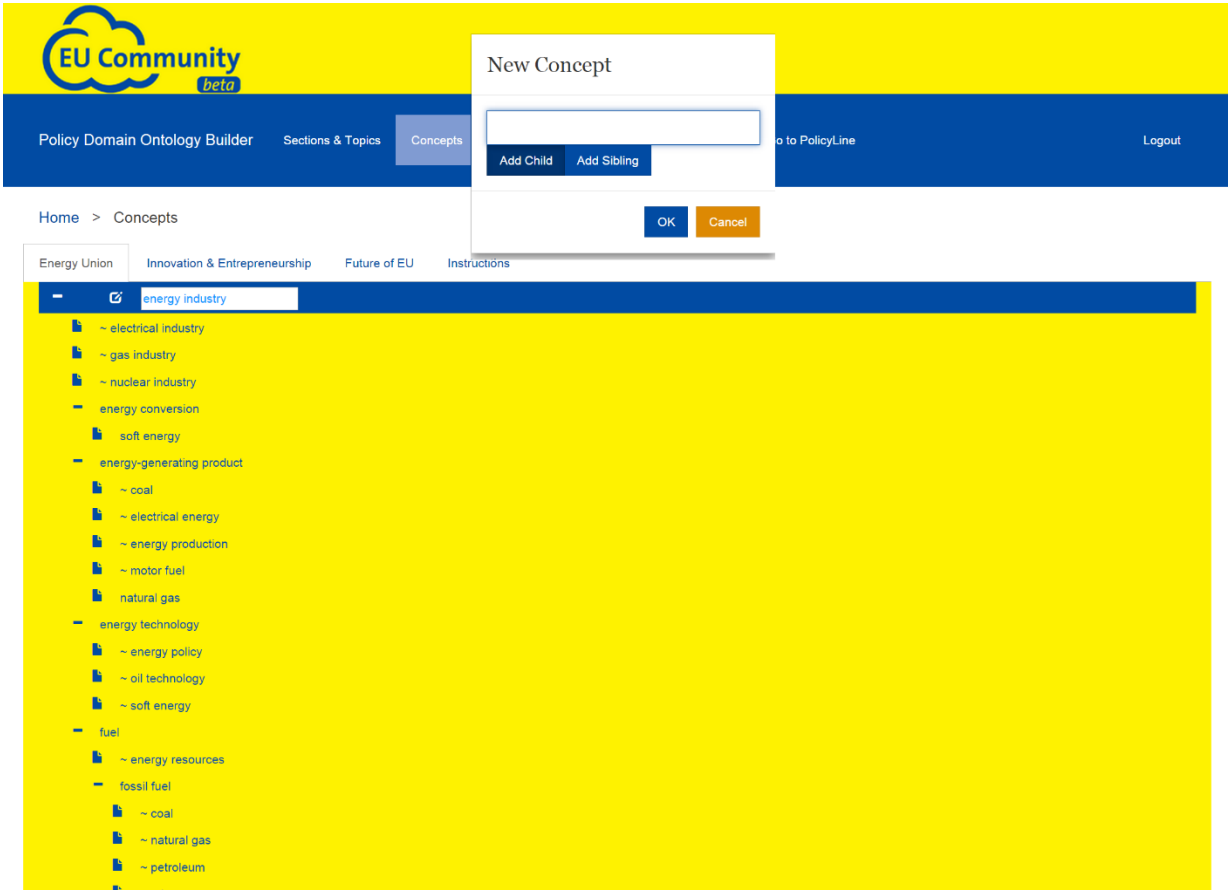


Figure 6: Concept Hierarchy Editor

3.4.2 Concept Hierarchies in the EU Community Project

The potential usefulness of these conceptual maps is multi-faceted:

1. It is possible to use the conceptual maps for performing ontology-based searches either in terms of index terms or of full-text search (using the concepts' primary and alternative textual representations).

2. A concept may be used for indexing documents coming from some source e.g. a document's metadata may include a list of relevant concepts this document is indexed under in its source; if an identical or encompassing indexing scheme happens to be employed by the EU Community platform, the document indexing metadata may be reliably carried over to the EU Community platform (this is a possibility we are actively pursuing for the case of EUROVOC as it is used in a number of data sources managed by the Publications Office, including EURLex, for which we have developed a crawler capable of retrieving information about legislative procedures.)

   Even for indexing schemes that are not guaranteed to be compatible, it may still be possible to carry over indexing information either in textual

form (terms/keywords) or by matching the source textual forms with EU Community's concepts' textual forms. The difference between being able to match document indexing terms on the level of concepts they refer to as opposed to textual matching of their representations is of qualitative importance, with the former scenario being obviously preferable due to the fact that it preserves the semantics of the indexing (something that will become particularly important if/when the EU Community platform supports multilingual content[4]).

3. Finally, the EU Community ontologies have the potential of becoming a useful resource in a number of text analysis tools in the EU Community platform and/or third-party tools.

3.5 Concepts-Topic Linking

3.5.1 Linking Concepts with Topics

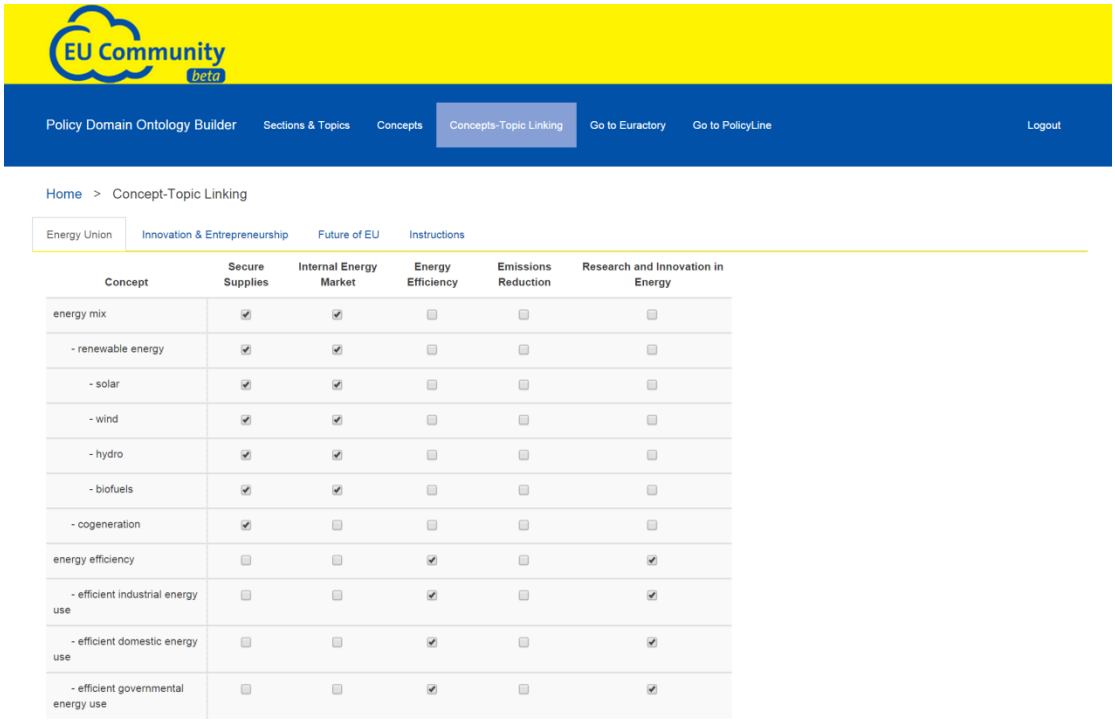
As mentioned earlier, it is conceptually correct to think that EU Community supports a single concept hierarchy, but for the purposes of keeping its maintenance manageable, the task of maintaining it is presented to users as a task of maintaining one concept hierarchy per Section. A side-effect of this design choice is that there is a simple relation linking Sections and Concepts: Concepts appearing in a Section's conceptual map may be assumed to be relevant to the Section (otherwise there would be no reason for them to appear in its conceptual map). The Policy Domain Ontology Builder provides functionality for also linking a Section's Topics with the Concepts in its conceptual map.

A user with the **ontology builder** role, logged in the Policy Domain Ontology Builder, may peruse and update the Concepts-Topics Linking Matrix by selecting "Concepts-Topic Linking" from the top options menu and then choosing the tab corresponding to a Section; the Section's Concept-Topic linking matrix will appear.

The interface is particularly simple:

- To link a Topic with a Concept, the user places a tick in the corresponding tickbox (the tickbox on the row of the Concept and the column of the Topic).
- To unlink a Topic with a Concept, the user removes the tick from the corresponding tickbox.

[4] Whereas this is not foreseen for the duration of the project, WP4 proceeds with the assumption that this will happen either in a potential follow-up project and/or when plans for commercial exploitation of the platform come to fruition.



Concept	Secure Supplies	Internal Energy Market	Energy Efficiency	Emissions Reduction	Research and Innovation in Energy
energy mix	☑	☑	☐	☐	☐
- renewable energy	☑	☑	☐	☐	☐
- solar	☑	☑	☐	☐	☐
- wind	☑	☑	☐	☐	☐
- hydro	☑	☑	☐	☐	☐
- biofuels	☑	☑	☐	☐	☐
- cogeneration	☑	☐	☐	☐	☐
energy efficiency	☐	☐	☑	☐	☑
- efficient industrial energy use	☐	☐	☑	☐	☑
- efficient domestic energy use	☐	☐	☑	☐	☑
- efficient governmental energy use	☐	☐	☑	☐	☑

Figure 7: Concepts-Topics Linking Matrix

3.5.2 Concept-Section and Concept-Topic Linking in EU Community

In Section 3.4.2, we outlined a number of possible uses of the Concepts in isolation from Sections and Topics and their links with them. Given that Sections and Topics are at the core of the content organization system of the EU Community platform, there is significant potential in the ontology that would remain untapped if the links between the Concept hierarchy and the Sections and Topics hierarchy did not exist. But links between Sections and Concepts exist because the Concept hierarchy is divided in per Section Concept hierarchies for usability reasons already explained and the Concept-Topics Linking Matrix is provided with the express purpose of recording the links between Concepts and Topics. Thanks to these links:

1. It is possible to move from Concept-based indexing of documents to document classification, namely associating documents (and legislative procedures) with Sections and Topics (the Sections and Topics linked with the Concept under which they were indexed). (This technique is to be used by the EurLex legislative crawler.)

2. It is possible to use the concept hierarchy's relations and textual representations as an additional input to a text classifier working with the full text of a document (rather than pre-defined indexing terms) aiming associate to it with the appropriate EU Community Topics and Sections.

3.6 Policy Processes

WP4 approaches the issue of 'policy modelling' from two very different angles. The first is Policy Domain Ontology Modelling; the outcome of relevant work includes the tools build to support building the policy domain ontologies (the Policy Domain Ontology Builder and Policy Domain Ontology Server) and the ontologies built with them as part of the project.

The second focuses on the concept of a policy process. Whereas the associated conceptual design is an important contribution of the present work package to the project (in tight collaboration with WP7), there is no associated WP4 software component that is responsible for creating, updating and perusing policy processes; these tasks befall the PolicyLine component (WP5), the main user-facing component of the project. In the first version of the Policy Component, the policy process and legislative procedure modelling undertaken in WP4 were only evident in the Predictions Module. The current, second version of the Policy Component introduces:

1. The EU Community Policy Process Ontology
2. The Policy Process Semantic Web Publication Agent

The latter is a sub-system that turns policy processes in CurActory into semantic web open data using a range of semantic web data representation languages and conforming to the EU Community Policy Process Ontology. That ontology in turn captures and formalises the conceptual modelling that has been arrived at as a result of Consortium-agreed modifications on the initial WP4 policy process model design.[5]

The policy process concept is the most central and important concept of the EU Community project. A **policy process** is the process through which various ideas and proposals addressing one or more aspects of EU policy are discussed, a legislative procedure is initiated on the basis of a proposal by an EU institution, the policy proposal becomes policy, and the impact of the policy is assessed.

A policy process may be lacking one or more of the above mentioned steps. For instance, a public discussion of proposals may not lead to the initiation of a legislative procedure by the EU institutions and a legislative procedure may lead to the rejection and/or withdrawal of the proposal that initiated it rather than result in new EU policy. From a practical viewpoint, recognising the beginnings of a policy process and creating an entry for it in the EU Community Platform is a difficult task that can only be carried out with the help of policy experts. So is the fragmentation of a policy process into discrete stages.

A **legislative procedure** is a formally defined procedure for the adoption of policy proposals as EU legislation, in accordance with the EU Treaties. Legislative procedures retrieved from EURLex have a title and other relevant metadata such as the type of the legislative procedure plus a list of **legislative procedure events**, in chronological order, each with a title (e.g. Adoption of the Commission

[5] In accordance with the original design, a policy process had a number of steps and documents were associated with those steps. The main difference in the current design is that this is no longer true. This simplifies data entry and allows the chronological disassociation of document authorship with process steps.

of a proposal, Formal Adoption by the Council etc.) and a date and possibly some associated documents (e.g. the original proposal with which the legislative procedure was initiated, an opinion on that proposal etc.).

Overall, the way we have decided to model policy processes is very flexible, serves the purposes of the Visualisation Component (PolicyLine) and is compatible with the modelling of legislative procedures on EurLex.

This last point is not without special importance. It means that a legislative procedure from EurLex can be embedded in a policy process; more specifically, given a policy process, it is possible to link to it a legislative procedure, whose documents will respectively become document lines (added to its existing document lines) and documents of the policy process and be automatically updated by a dedicated crawler, the EurLex legislative procedures crawler.

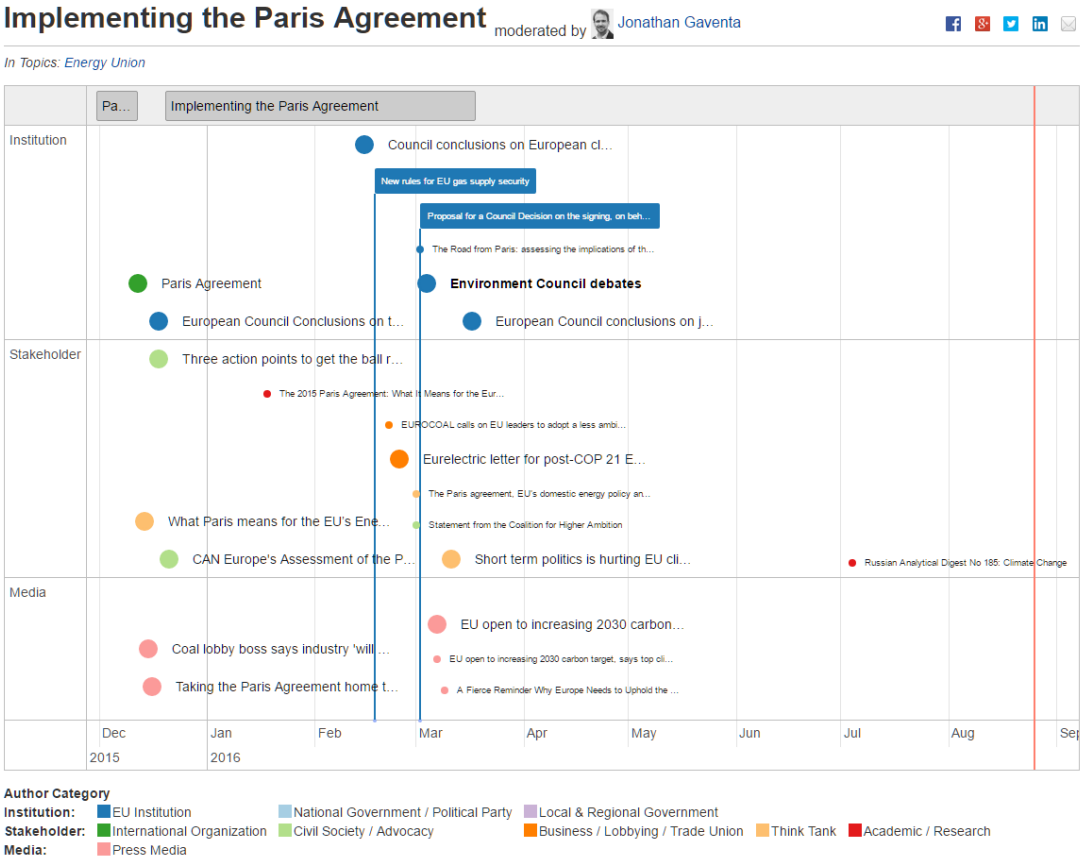


Figure 8: An example visualisation of a Policy Process

Below we note the main points of the policy process conceptual model as captured in the database design and meant to be recorded in the ontology.

- A policy process may be broken down into a number of distinct stages/periods; each period has a title, a start and an end date and the end date of one is (typically) the start date of the next one.
- A policy process may contain any number of documents; for each document, the following information is typically available:

- Title
- Author(s) (individuals and/or organisations)
- URL
- Category
- Status
- Date Created
- Date Submitted
- Submitted By
- Natural Language Processing-Derived Values
    - Sentiment
    - Objectivity
- Community Evaluation Scores
    - Accuracy
    - Value
    - Relevancy
    - Timeliness
    - Agree on Issue
    - Agree on Solution

- A policy process may contain a legislative procedure; the legislative procedure may have
    - A title
    - A URL
    - Outcome Predictions
    - Duration Predictions

Note: Importing a legislative procedure into a policy process via PolicyLine results in its documents becoming documents of the policy process.

The ontology developed as part of D4.3.2 can be found in Appendix B. The main effort in its design was not for it to be complete, but rather to be a solid foundation for future expansion. Neither the ontology nor the conceptual model are in any sense complete and final. This project has opened the discussion of how actual policy processes may be delineated and modelled, not closed it. It has given one possible conceptual framework, not *the* conceptual framework. It is likely that some of our current ideas will be retained in future projects, but it is equally likely that researchers, ourselves included, will venture into different directions also.

4 Predictions Module

4.1 Scope

The Predictions Module attempts to provide future progress estimates on the basis of known data, predictive modelling and user-provided knowledge or assumptions. It consists of two largely independent subsystems.

- The Hybrid Predictions Subsystem combines expert knowledge with generic predictive algorithms and statistical analysis, in order to answer specific questions regarding legislative procedures, such as:

  (a) Will the current proposal of a specific legislative procedure pass without amendments, with amendments or not at all?

  (b) When will the specific legislative procedure end?

  In order for the Hybrid Predictions Subsystem to be able to make these predictions, it uses (a) historical data from the EURLex Legislative Procedure crawler (b) any available expert user input (collected by PolicyLine).

- The Simulation Subsystem attempts to predict the timeline progression of quantitative metrics such as awareness, engagement, and sentiment relating to policy processes and their associated documents (relying on user provided content, crawled data, and the Opinion Mining component). The difficulty of making reliable predictions about anything non-trivial is well understood. The main aim of the Simulation Subsystem is to predict the overall direction of the dynamics of a policy process-centered discussion.

4.2 User Roles

There are currently no restrictions on the use of either the Hybrid Predictions Subsystem or the Simulation Subsystem by any user of the EU Community.

4.3 Policy Processes and Legislative Procedure Modelling in the Predictions Module

As mentioned in Section 3.6, one of the two ways WP4 approached the issue of 'policy modelling' was the creation of the conceptual design of a policy process which is central to the EU Community project and accessible to users via the PolicyLine component. This conceptual model has been formalised as the EU Community Policy Process Ontology which in turn serves as one of the foundations of the design of the Policy Process Semantic Web Publication Agent.

However, Policy Process Modelling and Legislative Procedures modelling are also very relevant to the Predictions Module. Legislative Procedure modelling is the basis of the Hybrid Predictions Module design and Policy Process modelling is at the core of the predictive model of the Simulation Subsystem.

There is one difference that is worth mentioning.

Policy process modeling is very relaxed and minimalistic in terms of internal structure and constraints, at least at the technical level (the project experts and the project pilots have guided us in the direction of pre-defining some of the policy process structure for greater homogeneity, but any decisions made are not cast in stone and may change at any point).

Legislative procedures, on the other hand, are very rigidly defined with predefined and limited degrees of freedom in their evolution, as they are governed by specific rules (e.g. the Treaties that created the European Union and its legislative framework). Their complex rules are not captured in the EU Policy Process Ontology as it is not the EU Community Platform that controls the legislative workflows, but rather the responsible institutions; yet, the EURLex legislative procedure data that is imported to the EU Community platform follow these rules and this makes such data easier to work within certain respects.

This difference, together with the fact that there is an data source where thousands of legislative procedures are available as open data (EURLex), is what enabled us to create the Statistical Predictor for Legislative Procedures, which later became the basis for the Hybrid Predictions Sub-System.

Because the Simulation Subsystem is more concerned with the characteristics of the mix of documents relevant to whatever it is that it provides predictions about, it can work with any kind of policy process, with a legislative procedure, or even an amorphous collection of documents on the same topic. Because the past data-based component part of Hybrid Predictions Subsystem relies on filtering relevant past data on the basis of pre-defined categories of legislative procedures and legislative procedure events, it works well with the event sequences of legislative procedures, but it cannot work with an open-ended definition of a Policy Process. Because a legislative procedure can be part of a policy process, the Hybrid Predictions Subsystem is relevant such policy processes also, but only for the legislative procedure part.

4.4 Hybrid Predictions Subsystem

4.4.1 Why a Hybrid Predictions Subsystem?

The Hybrid Predictions Subsystem aims to make the best possible use of two sources of information for predicting how a legislative procedure will unfold: past legislative procedure data and user predictions. It owes much to a very interesting discussion between the review panel and consortium members during the EU Community project's Year 1 Review Meeting and combines features that an expert input-based system and a statistics-based system would (further ideas for refinements discussed in the Year 2 Review Meeting and re-iterated in the Year 3 Interim Review Meeting will be addressed in D4.4.2.)

What was originally envisaged and described in D6.2 was a system that would only use past data in order to predict the future. The pertinent evaluation questions regarding such a system were:

- How accurate would the predictions be?
- How would it compare to a human expert's predictions?

In view of the overarching approach of the project to attempt to provide a blended approach combining the best of what the current state of AI can bring and the best of what a community of experts can bring, it was suggested that experts input should also be used. Far from providing an easy way of avoiding a comparison with human experts, the proposal to combine past data with experts' input made it all the more pertinent and additionally created a new set of challenges for the project. Practically all these new challenges stem from the fact that the system will need to make predictions in cases where the experts' predictions differ and/or are not in agreement with the prediction that would have been made by looking at past data.

These challenges were welcomed by the WP4 technical partners for one very specific and concrete reason: whereas past data can provide probabilities of outcomes regarding a specific legislative process, they are blind to what the content of the relevant proposal is and the dynamics of support and opposition that an expert in the know will be privy to. Experts can offer predictions based on the specific process, whereas past data can only offer statistics about past processes.

4.4.2 Introducing Prediction Games

The idea to have an approach combining what algorithms do best with what human experts know is appealing. But are all users true experts and are their predictions reliable? Do the carry equal weight? How can opposing predictions by experts and predictions by the EU Community Platform's AI be combined into a single prediction (by the platform's AI)? The promise of better results by also using expert input is but a promise. Much needs to be done to make it happen.

The answer given by WP4 to these challenges was to use a model based on game concepts to

(a) provide a framework for the reliability of users' abilities to be assessed on the basis of the success or failure of their past attempts to answer questions successfully

(b) provide a framework in which any attempts to define AI-based prediction making agents can participate on the same terms as human experts (resulting also in automatic evaluation)

For this approach to produce reliable predictions, the predictions assessment rules should be such that any one player (and that could indeed be the/an AI player) can make a prediction that will be valued higher than the predictions even of all other players (should that one player's performance be higher than the sum of the performances of those other players); it should be the collective weight of those siding with one prediction that needs to be taken into consideration when comparing it with another supported by a different set of players, not the mere number of supporters.

Such an approach has the additional benefit that it allows not just one but potentially more ways of trying to utilise the available data for prediction making to be tried and evaluated; a number of AI players can be defined each implementing a different prediction methodology. Indeed, that provides, as a bonus, way to comparatively evaluate them.

What is described here is a kind of under-the-hood gamification that may or may not reach the user. In other words, it is entirely possible to keep all the game-

based terminology and conceptual model for combining predictions by different experts and by different algorithms, as well as their scores hidden from the users. This is but one option. It is also possible to encourage users to make predictions by showing them their score and sending them notifications when on the basis of updates on the status of legislative procedures on which they have made predictions they gain or lose points. The predictions-based score may also be made into an additional criterion on the Reputation Management System.

These are options that have be discussed by the Consortium. The decision was to attempt to implement a Predictions criterion in the RMS, as a joint effort between WP3 and WP4. This decision does not preclude any further developments on this front in follow-up projects or as part of the current project's exploitation attempts.

One consequence of the decision to move towards utilizing user input is that user interface design and questionnaire design become concerns for the design of the Hybrid Predictions Subsystem. The two most important prediction question that users may be interested having answered or answering themselves have been chosen:

(a) Will the current proposal of a specific legislative procedure pass without amendments, with amendments or not at all?

(b) When will the specific legislative procedure end?

Each question defines a separate kind of game and will be treated separately for scoring purposes. That means that any experts and/or AI agents will be evaluated separately on their ability to predict outcomes and times. Also as AI agents are based on certain algorithms and these will differ for different questions, different AI agents will be attempting to answer different questions.

4.4.3 The Final Outcome Prediction Game

The players are the users of PolicyLine plus a number of AI agents defined below. All players start out with 0 points and may gain point by participating in games. Each active legislative procedure is a game and each active legislative procedure stage is a round. Each player may choose to play in as many games as they are interested in and, for each game, in as many rounds as they wish. Participating on a round involves answering the following question:

What do you think the legislative procedure's outcome for the current policy proposal?

The possible answers are:

1. It will not pass at all (i.e. it will be either rejected or withdrawn)
2. It will pass with amendments
3. It will pass with no amendments

Given the nature of this question, players will only know if they have won or lost on a round they have played in after the end of the game (the conclusion of the legislative procedure).

For each answer, its value v at the time the round ended is computed (see below). For each player that answered it correctly, v/c points are added to their

score (where *c* is the number of players that answered it correctly). For each player that answered it incorrectly, *v/i* points are subtracted from their score (where *i* is the number of players that answered it incorrectly).

The value of an answer is 100 × (1-p) where p is the probability of the answer on the basis of past data statistics, computed according to the COMPUTE_PROBABILITY_OF_OUTCOME algorithm below. The most distinguishing feature of the algorithm is the use of sequences consisting of the legislative procedure type, the stages it has been through and the decision taken at each, if any. The idea is that the probability of a certain outcome may depend on the type of legislative procedure and on its history. As the case is with such attempts to select the most pertinent data, the more a legislative procedure progresses the fewer the matching past legislative procedures to be considered will be; if that proves to be a problem a fall-back strategy will be devised.

COMPUTE_PROBABILITY_OF_OUTCOME
Parameters:
L : a legislative procedure
O : possible outcome for L's current proposal
(one of Pass, Pass with Amendments, Fail (Rejection or Withdrawal))
Output:
p: the probability of O given the current progress in L

Step 1. Create a sequence consisting of the following form:
$\tau:\sigma_1:\delta_1 : \ldots :\sigma_n:\delta_n$ where
τ is the type of the legislative procedure *L* (e.g. Ordinary Legislative Procedure)
$\sigma_{i\ (1<=i<=n)}$ is the Stage *i* Name (e.g. EP Opinion on 1st Reading)
$\delta_{i\ (1<=i<=n)}$ is the Stage *i* Decision (e.g. Approval with amendments) (or - if N/A)
n is the most recent stage of *L*.

Step 2. Filter past completed legislative procedures, keeping only those with a sequence (constructed as above) that is a prefix of the current legislative process' sequence

Step 3.
Let P = the number of matching legislative procedures whose proposal passed without amendments
Let A = the number of matching legislative procedures whose proposal passed with amendments
Let F = the number of matching legislative procedures whose proposal did not pass

Step 4. Return P/(P+A+F),A/(P+A+F), or F/(P+A+F) depending on the value of *O (Pass, Pass with Amendments or Fail).*

Combining the opposing predictions of users was not the only motivation for the game-based design of the Hybrid Predictions Subsystem; it is termed `hybrid' because it would not be either entirely data-based or entirely expert opinion-based. We have discussed about human players; the missing ingredient is the AI players.

As was mentioned earlier, the original intent was not to have a Hybrid Predictions Subsystem but a statistics-based one. The originally envisaged design can be used to define an AI player. As a consequence, this method's performance, now

rendered as an AI player, can be measured against the performance of human experts.

Statistics-Based AI Player
Let P = the probability for the outcome Pass according to the COMPUTE_PROBABILITY_OF_OUTCOME algorithm Let A = the probability for the outcome Pass with Amendments according to the COMPUTE_PROBABILITY_OF_OUTCOME algorithm Let F = the probability for the outcome Fail according to the COMPUTE_PROBABILITY_OF_OUTCOME algorithm Choose the answer Pass, Pass with Amendments, Fail on the basis of which of P, A or F is greatest.

It is also possible define an AI player that ignores past-input statistics but rather relies only on the predictions of users. This can be used for evaluating, within the given gaming framework, the input from the platforms users as a collective.

User-Input-Based AI Player
Let P = the sum of the scores of all human players who predicted the outcome Pass Let A = the sum of the scores of all human players who predicted the outcome Pass with Amendments Let F = the sum of the scores of all human players who predicted the outcome Fail Choose the answer Pass, Pass with Amendments, Fail on the basis of which of P, A or F is greatest.

Finally, it is also possible to define a game-based AI Player to participate in the game. The aim of defining the Game-Based AI Player is to allow the game-based approach to be evaluated within the given gaming framework.

Game-Based AI Player
Let P = the sum of the scores of all other players who predicted the outcome Pass Let A = the sum of the scores of all other players who predicted the outcome Pass with Amendments Let F = the sum of the scores of all other players who predicted the outcome Fail Choose the answer Pass, Pass with Amendments, Fail on the basis of which of P, A or F is greatest.

While we have addressed issues related to the concept of games which we have introduced, such as how scores will be affected on the basis of predictions and actual outcomes, we have not addressed the original question: how the Hybrid Predictions Subsystem will decide which prediction to produce on the base of

conflicting predictions. The answer the entire discussion has been leading to is this: the Hybrid Predictions Subsystem will return the outcome predicted by the Game-Based AI Player.

4.4.4 The End Date Prediction Game

The details of this game are similar to the details of the Final Outcome Prediction Game already discussed. The main difference is the question:

When do you think the legislative procedure will be concluded?

The possible answers are the current month and months in the future.

The possibility of each answer is computed in terms of how long it predicts the legislative process will have taken from start to finish.

COMPUTE_PROBABILITY_OF_DURATION Parameters: *L : a legislative procedure* *M : possible duration of L in months* *Output:* *p: the probability of M given the current progress in L*
Step 1. Create a sequence consisting of the following form: $\tau:\sigma_1:\delta_1 : \ldots :\sigma_n:\delta_n$ where τ is the type of the legislative procedure *L* (e.g. Ordinary Legislative Procedure) $\sigma_{i\ (1<=i<=n)}$ is the Stage *i* Name (e.g. EP Opinion on 1st Reading) $\delta_{i\ (1<=i<=n)}$ is the Stage *i* Decision (e.g. Approval with amendments) (or - if N/A) n is the most recent stage of *L*. Step 2. Filter past completed legislative procedures, keeping only those with a sequence (constructed as above) that is a prefix of the current legislative process' sequence and a total duration that is the same as the current duration of *L* or longer. Step 3. Return the count of filtered legislative procedures which were concluded in M months divided by the count of all filtered legislative procedures.

Three different approaches to making legislative procedure duration predictions similar to the ones considered for the problem of predicting a legislative procedure's final outcome can be defined here also as AI players as was done for the Final Outcome Prediction Game.

The first is based on the originally envisaged methodology for providing a statistics-based prediction.

Statistics-Based AI Player
Choose the month corresponding to adding the most probable duration of the legislative procedure (according to the COMPUTE_PROBABILITY_OF_DURATION algorithm) to its start data.

It is also possible define an AI player that ignores past-input statistics but rather relies only on the predictions of users.

User-Input-Based AI Player
Choose the most heavily supported prediction regarding the legislative processes end date (on the basis of the sum of scores of players making each prediction) among human players' predictions.

The same simplistic approach is carried to the third AI player.

Game-Based AI Player
Choose the most heavily supported prediction regarding the legislative processes end date (on the basis of the sum of scores of players making each prediction) among the predictions of all other players.

As was the case in the Final Outcome Prediction Game, the Hybrid Predictions Subsystem will produce the answer given by the Game-Based AI Player as its own prediction.

D4.4.2 describes the steps taken to set up an evaluation framework and to improve on the initial algorithms.

4.5 Simulation Subsystem

The Simulation Subsystem focuses on policy processes. It is responsible for providing future projections, as prescribed in deliverable D4.2. It utilizes data in the EU Community platform as the basis of estimations on the evolvement of the debate around a policy process (e.g. awareness, engagement, controversy).

The operation of the Simulation Subsystem lies on a simulation mechanism, i.e. a simulation model, which runs in the back-end to provide results predicting how a policy process, and more specifically the set of chosen policy process metrics will evolve over time. For this task, the System Dynamics methodology has been applied.

The way the Simulation Subsystem is designed to operate and interoperate with the other components of the EU Community Platform is outlined below:

1. The Simulation Subsystem takes as input information from CurActory, the common database, including the stored results of analyses and computations by the Opinion Mining & Reputation Management Component

2. The Simulation Subsystem processes the retrieved information through the underlying modelling and simulation mechanism.

3. The Simulation Subsystem outputs the simulation output.

4.5.1 Model Definition

In a nutshell, the model's objective is to provide time series predictions as an overview on the current status and future progress on the policy process

associated with a policy topic. Therefore, our goal was to create a model that simulates the aspects related with a policy process using a set of quantitative metrics able to provide an overview of its status.

To achieve that, we introduced a set of metrics that can be derived from the available EU-Community data and are able to quantify the different levels of the engagement and participation in the policy process. These are listed in the following table.

Table 2: The policy process metrics

Metric	Definition
Sentiment	Positive - Negative Documents
Controversy	Positive × Negative Documents
Volume	Positive + Negative Documents
Awareness	The number of people aware of the policy topic
Engagement	The number of people engaged in the policy deliberation process

The sentiment metric is a simple equation showing the overall public sentiment as calculated by the documents related to the policy process (the positive documents minus the negative ones. If the negative documents are more than the positive then the sentiment metric takes a negative value. The controversy metric shows the level of homogeneity and polarity between positive and negative sentiments. In case there are many positive and negative documents, the metric tends to increase, otherwise it remains low; if there are only negative or only positive documents controversy is zero. The Volume metric simply measures the number of positive and negative documents. Awareness on the policy process is a metric concerning the number of people that have read about it, whereas in our model Engagement is a metric concerning the number of people that have performed any short of non-passive activity related to the Policy Process; e.g. written about it, evaluated documents etc.

Regarding the available input, the calculation of the above metrics will be based on data on the key entities of the EU-Community database, i.e. documents and persons identified as being relevant to EU policy making. Other aspects, such as events or planned activities will be incorporated through users' input.

4.5.1.1 Causal Loop Diagram

A preliminary step for building the model was to capture the correlations between the aforementioned metrics in a casual loop diagram. This is the basic tool of System Dynamics methodology to capture the qualitative influence between the system variables. To identify the core elements of the model, we focused on the two core concepts of the project, namely **documents** and **persons.**

With respect to the documents related to a policy process or a topic it can be safely assumed that documents generate other documents, regardless what type these documents are (media reports, citizen's interaction in social media) etc. And the more documents are generated, the more engagement is induced among people. Taking into account the word of mouth effect and the social media viral dissemination phenomenon, we make the assumption that as documents increase, more documents are being produced.

But, documents rate cannot increase forever. There should be saturation, so that as time passes, documents production decreases. In terms of System Dynamics, we refer to internal feedback loops of our system. This can be seen in the following figure. The same principle was implemented in all components of the model.

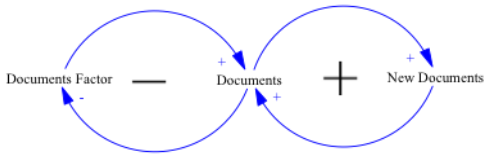


Figure 9: Document generation loop

Another characteristic of the project is the sentiment classification, which can be depicted through the separation of documents into positive and negative in the model. Another assumption we make, is that if the rate of positive documents is big, then the rate of negative documents cannot behave in the same way. This can be seen in, where as positive documents increase, the rate of negative documents decreases and vice versa.

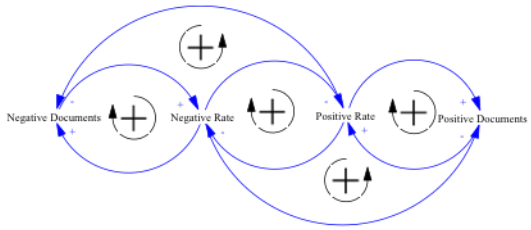


Figure 10: Positive and negative documents variation

The above assumptions have been adopted in the casual loop diagram presented below, which captures all system's parameters and the linkages between them. The arrows in the following schema illustrate the relationship between each of the factors involved in the behavior of the system.

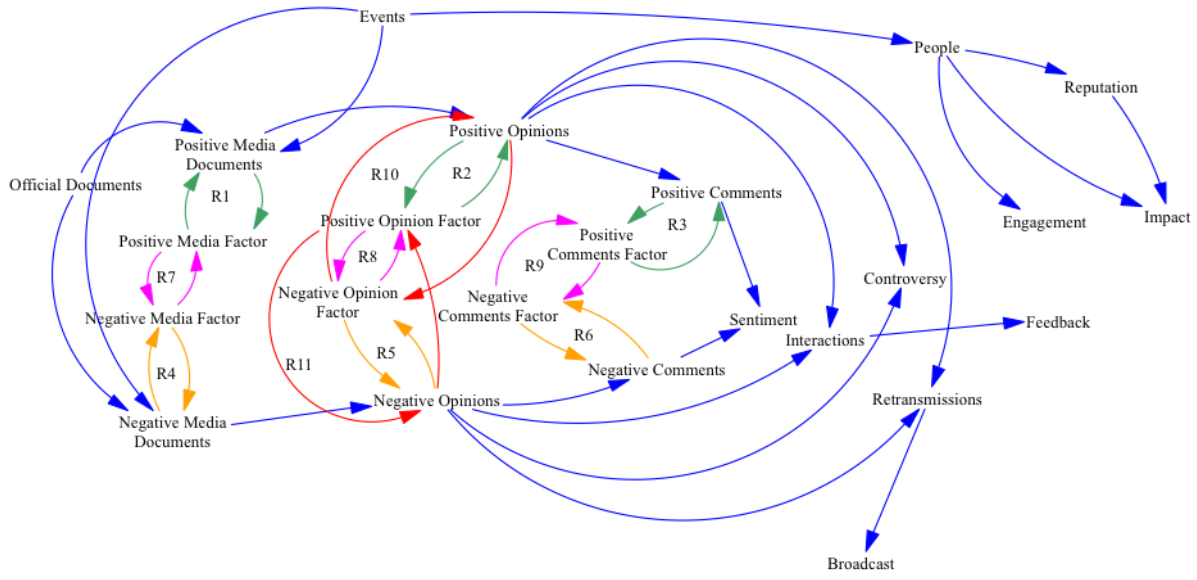


Figure 11: The Causal Loop Diagram of the simulation model

Another reinforcing factor is the events variable, which determines if a relevant public event is held, which is possible to initiate an increase of the people engaged with the discussion topic. All these variables are aggregated to the metrics defined to measure the evolution of the policy process.

4.5.1.2 Stock and Flow Diagram

The final step was the creation of the stock and flow diagram. This is the quantitative representation of the simulation model (provided in Figure 12), as opposed to the causal loop diagram, which is a qualitative representation. Therefore, in this task, starting from the causal loop diagram we implemented the functions and equations that are used to translate the model in a mathematical way, which enables appropriate computer software to run simulation scenarios on it.

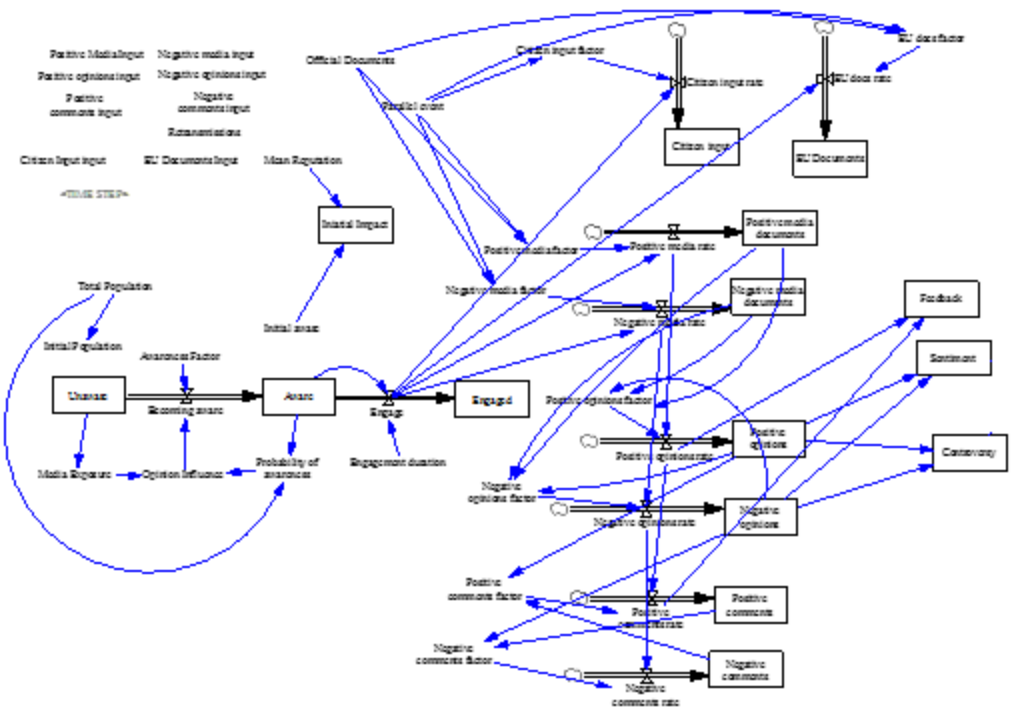


Figure 12: Stocks and Flows Diagram

A System Dynamics model is made of four kinds of elements: Stocks, Variables, and Flows. A **Stock** is a collection of stuff, an aggregate. For example, a Stock can represent the number of documents retrieved. A **Flow** brings things into, or out of a Stock, based on defined equations. Flows look like pipes with a faucet because the faucet controls how much stuff passes through the pipe. A **Variable** is a value used in the diagram. It can be an equation that depends on other Variables or on user's input, or it can be a constant. The objective of our System Dynamics model is to figure out how the value of Stocks changes over time by estimating them over and over. Therefore, the requested indicators to be monitored and predicted were represented as stocks in the model. Some of them have been decoupled to other stock entities, e.g. Documents comprise of Positive and Negative Documents. Based on the System Dynamics modelling principles, the identified system parameters were connected as variables, stocks and flows of the model and the functions determining their behavior.

For the design of the model, Vensim PLE[6] was utilised, since it meets the integration requirements prescribed by PySD, as described in the next sub-section. Vensim is a simulation tool that facilitates the development of continuous and discrete simulation models, supporting graphical and textual model construction.

4.5.2 Implementation and Integration

There are multiple layers in the implementation of the Simulation Subsystem:

[6] www.vensim.com

1. At its very core is the Vensim System Dynamics model which determines its behaviour and results.

2. The execution engine is implemented in Python on the basis of PySD[7], a dedicated System Dynamics package offered by the Python community. The PySD library allows the integration of Vensim SD models in Python, out of the box. The major advantage of this approach is that just improving the model and substituting the model file, the Simulation Subsystem can be upgraded very easily.

3. The integration layer is provided by an ASP.NET Web API which will be responsible for receiving requests with user-defined parameters from PolicyLine, collecting model initialisation data from CurActory, invoking the model execution engine, retrieving the results and passing them back to PolicyLine as a response to its request.

[7]PySD documentation, https://pypi.python.org/pypi/pysd/0.1.0

5 Conclusions – Future Work

A lot of work has been carried out in different directions in order to produce the first and second prototypes of the Policy Component.

1. A very efficient Policy Domain Ontology Builder tool was created on the basis of novel ideas such as the use of a micro-language to encode states and relations between Concepts as well as alternative textual representations. The same tool also supports the curation of the Sections and Topics hierarchy (including the approval of user-suggested Sections and Topics) as well the linking of topics and concepts.

2. A semantic ontology for policy processes formalising the conceptual design that informed the EU Community database schema design on the one hand and the PolicyLine visualisation design on the other was created.

3. A web-based mechanism for exposing EU Community policy processes as open data was added to the EU Community Platform's arsenal.

4. An innovative and readily expandable Hybrid Predictions Subsystem was designed that allows predictions about legislative procedures to be made on the basis of not only statistics on past relevant data or expert opinions but of a combination thereof. Its innovation lies in the use of a games-based approach to the challenge of choosing between contradictory predictions derived by different algorithms or produced by different experts. Not only does it leave open the possibility for numerous alternative ways of making automated predictions on the basis of data available to the EU Community Platform but it also provides a framework for their evaluation.

5. A Simulation Subsystem based on a System Dynamics model was created. The model was created on the basis of assumptions about the effect of the publication of one type of document may have on other factors motivating the creation and publication of documents. It is worth noting that whereas the System Dynamics methodology has been well-documented and used in a number of fields, to our knowledge there are no models approaching any kind of policy process (the most relevant extant models deal with general aspects of online participation).

Further WP4 Work

It is best to think of D4.3.1 as a specification for the relevant components of WP4 and of D4.4.1 as a deliverable stating what one learns from evaluating (with the benefit of real data) what seemed to make sense on paper as a specification in the preceding deliverable. D4.3.2 is the second and final version of the WP4 components' specification; it defines the functionality the second and final version the Policy Component prototype is meant to deliver.

Therefore, D4.3.2 and D4.4.2 mark the end of the line for a very challenging Work Package in a very interesting and ground breaking project. D4.4.2 will present the improvements that were made possible by having evaluation targets and test data to support these evaluations, as well, of course, as the final evaluation results for the Policy Component.

Exploitation

Moving forward, what is of the essence is that this project's results be exploited and that it lives on be in as part of a commercialisation effort, as part of a project furthering its ideas and ambitions, or a combination thereof.

Clearly, the future of EU Community project is tied with the success for its commercial and research exploitation. Interestingly, this focus, evident also on the fact that an interim Y3 review meeting largely dedicated to exploitation was deemed necessary and indeed has proven beneficial, is also reflected in the final WP4 deliverables, as will be discussed in more detail in D4.4.2. With the addition of the Policy Process Semantic Web Publication Agent and the EU Community Policy Process Ontology, the Policy Component prototype is both more complete and serves the goals of the project better, especially with respect to its commitment towards open data. At the same time though, the updated second version of the Policy Component prototype also serves well the involved partners' future exploitation plans as it not only provides functionality to the EU Community Platform, but also covers a number of research and development areas where both the expertise and the software developed can be assets suitable for exploitation beyond the current project and any exploitation efforts focusing on it.

APPENDICES

Appendix A: Integration Web APIs

A.1 Policy Domain Ontology Web API

The Policy Domain Ontology Server (henceforth the Server) provides an interoperable RESTful Web API that supports the functionality of the Policy Domain Ontology Builder (henceforth the Client). It is a rather unusual API because it is aimed to cater both for the possibility of a lost connection and for the possibility of multiple users working concurrently on the ontology.

A.1.1 Protocol

- The Client's first request upon initialisation is a GET request to the URL "/api/action?t=topics". The response is a JSON-formatted list of action messages that if `executed' by the client will construct the current Sections & Topics hierarchy. The final message in the list is a SET_VERSION action message.
- Subsequent messages from the Client describing any user actions affecting the Sections and Topics hierarchy will be POST requests to the URL "/api/action?t=topics&v=**N"** where **N** stands for version number last communicated by the server using a SET_VERSION action message. The body of the message will normally be a list containing a single action message, the one describing the latest user action on the Sections & Topics hierarchy. However, if the Client has a backlog of action messages for which it has not yet received a response from the server, it includes the corresponding action messages in the list in the order the corresponding actions had been performed. The response from the server is a list of action messages concerning actions performed by other users followed by a SET_VERSION action message.
- It becomes clear that action messages play a central role in both the Client and the Server. In fact every user action in the client is immediately converted to an action message, processed by the action message interpreter, stored in the backlog of unacknowledged messages, and sent asynchronously to the Server (the success callback set to an action that removes it from the unacknowledged messages backlog and informs the user that the action has been committed).
- The Server stores for each action message it receives the Client Perceived Version, the User, the SessionId, ActionMessage, Resulting Version. If the Server receives again the same ActionMessage with the same SessionId and the same Client Perceived V, it ignores it.
- The Client's first request upon the user selecting to edit a Section's Concepts hierarchy or perform Topics –Concepts linking is a GET request to the URL "/api/action?t=concepts&section=**S**" where **S** is the id of the section. The response is a JSON-formatted list of action messages that if `executed' by the client will construct the current Concepts hierarchy for

the topic. The final message in the list is a SET_VERSION action message. The Client may decide not to send the above request if it was previously in a context of editing the same section's hierarchy.

- The Client's second (or parallel) request upon the user selecting to perform Topics –Concepts linking is a GET request to the URL "/api/tl?section=**S**" where **S** is the id of the section. The response is a list of list the first element of which is a Concept id and the second is a list of the Topic ids (of the given Section) it is connected with.
- Subsequent messages from the Client describing any user actions affecting the Concepts hierarchy and Topic-Concept linking will be POST requests to the URL "/api/action?t=concepts&section=**S**&v=**N"** where **S** stands for the section displayed in the user's interface and **N** stands for version number last communicated by the server using a SET_VERSION action message. The body of the message will normally be a list containing a single action message, the one describing the latest user action on the Section's Concepts hierarchy. However, if the Client has a backlog of action messages for which it has not yet received a response from the server, it includes the corresponding action messages in the list in the order the corresponding actions had been performed. The response from the server is a list of action messages concerning actions performed by other users on the same Section's Concepts hierarchy (and its Topic-Sections linking) followed by a SET_VERSION action message.

A.1.2 Sections and Topics Action Messages

ADD_SECTION_OR_TOPIC		
Input Parameter 1 Name ParentId	**Input Parameter 1 Type** LONG	**Input Parameter 1 Documentation** The id of the section under which the new Topic will be added; if ParentId is null, the action concerns a Section rather than a Topic.
Input Parameter 2 Name Definition	**Input Parameter 2 Type** STRING	**Input Parameter 2 Documentation** The definition of the new Section or Topic.
	Output Type LONG	**Output Documentation** The id of the newly added Topic or Section.

MODIFY_SECTION_OR_TOPIC

Input Parameter 1 Name	**Input Parameter 1 Type**	**Input Parameter 1 Documentation**
`Id`	LONG	The id of the Section or Topic.
Input Parameter 2 Name	**Input Parameter 2 Type**	**Input Parameter 2 Documentation**
`Definition`	STRING	The modified definition of the new Section or Topic.

A.1.3 Concept-Related Action Messages

NEW_CONCEPT		
Input Parameter 1 Name	**Input Parameter 1 Type**	**Input Parameter 1 Documentation**
`SectionId`	LONG	This is the id of the Section of the ontology in which the addition of the newly created concept is to be performed.
Input Parameter 2 Name	**Input Parameter 2 Type**	**Input Parameter 2 Documentation**
`ParentConceptId`	LONG	The id of the concept under which the new concept will be added.
Input Parameter 3 Name	**Input Parameter 3 Type**	**Input Parameter 4 Documentation**
`Relation`	“” or “~”	The relation of the new concept to its parent.
Input Parameter 4 Name	**Input Parameter 4 Type**	**Input Parameter 3 Documentation**
`Definition`	STRING	The definition of the new concept to be added.
	Output Type	**Output Documentation**
	LONG	The id of the newly added concept.

ADD_CONCEPT		
Input Parameter 1 Name SectionId	**Input Parameter 1 Type** LONG	**Input Parameter 1 Documentation** This is the id of the Section of the ontology in which the addition of the newly created concept is to be performed.
Input Parameter 2 Name ParentConceptId	**Input Parameter 2 Type** LONG	**Input Parameter 2 Documentation** The id of the concept under which the concept will be added.
Input Parameter 3 Name Relation	**Input Parameter 3 Type** “” or “~”	**Input Parameter 3 Documentation** The relation of the concept to be added to its parent.
Input Parameter 4 Name ConceptId	**Input Parameter 4 Type** LONG	**Input Parameter 4 Documentation** The id of the concept to be added.

MODIFY_CONCEPT		
Input Parameter 1 Name ConceptId	**Input Parameter 1 Type** LONG	**Input Parameter 1 Documentation** The id of the concept to be modified.
Input Parameter 2 Name Definition	**Input Parameter 2 Type** STRING	**Input Parameter 2 Documentation** The modified definition of the concept.

REMOVE_CONCEPT		
Input Parameter 1 Name SectionId	**Input Parameter 1 Type** LONG	**Input Parameter 1 Documentation** This is the id of the section from the

		ontology for which the removal is to be performed.
Input Parameter 2 Name `ConceptId`	**Input Parameter 2 Type** LONG	**Input Parameter 2 Documentation** The id of the concept that is to be removed.

A.1.4 Concept-Topic Linking-Related Action Messages

LINK_CONCEPT		
Input Parameter 1 Name `TopicId`	**Input Parameter 1 Type** LONG	**Input Parameter 1 Documentation** This is the id of the topic with which the concept is to be linked.
Input Parameter 2 Name `ConceptId`	**Input Parameter 2 Type** LONG	**Input Parameter 2 Documentation** The id of the concept that is to be linked with the given topic.

UNLINK_CONCEPT		
Input Parameter 1 Name `TopicId`	**Input Parameter 1 Type** LONG	**Input Parameter 1 Documentation** This is the id of the topic with which the concept is to be unlinked.
Input Parameter 2 Name `ConceptId`	**Input Parameter 2 Type** LONG	**Input Parameter 2 Documentation** The id of the concept that is to be unlinked with the given topic.

A.2 Hybrid Predictions Subsystem Web API

***Question 1:* Will the current proposal of the policy process's legislative procedure pass without amendments, with amendments or not at all?**	
API Call	**GET /api/LegislativeProcedurePrediction/Outcome**

	Inputs (Passed in the Query String, URL Encoded)
	PolicyProcessId : Integer
	Output
	JSON-formatted string for one of the following: Pass, PassWithAmendments, Fail
	This Web API call takes the id of a policy process with a Legislative Procedure attached to it and returns an estimation of the outcome of that legislative procedure.

***Question 2:* When will the policy process's legislative procedure be concluded?**	
API Call	
	GET /api/LegislativeProcedurePrediction/Completion
	Inputs (Passed in the Query String, URL Encoded)
	PolicyProcessId : Integer
	Output
	JSON-formatted ISO-8601 date string
	This Web API call takes the id of a policy process with a Legislative Procedure attached to it and returns an estimation of the completion date of that legislative procedure.

A.3 Simulation Subsystem Web API

***Question:* Given its current state, how will the policy procedure progress in terms of the supported list of metrics in the next 6 months?**	
API Call	

	GET /api/PolicyProcessPrediction/Simulation
	Inputs (Passed in the Query String, URL Encoded)
	PolicyProcessId : Integer
	Output
	JSON-formatted object containing the following fields: • Unaware • Aware • Engaged • PositiveDocuments • NegativeDocuments • Sentiment • Controversy • Volume Each field has its value a list of 27 numbers.
	This Web API call takes the id of a policy process and returns a number of timeseries, as lists of numbers the first of which corresponds to the initial situation and each subsequent number containing the model's prediction for one week later.

Appendix B: Policy Process Ontology

B.1 Policy Domain Ontology

```
# baseURI: http://eucommunity.eu/ontology



@prefix euc: <http://eucommunity.eu/ontology#> .

@prefix dcterms: <http://purl.org/dc/terms/> .

@prefix foaf: <http://xmlns.com/foaf/0.1/> .

@prefix owl: <http://www.w3.org/2002/07/owl#> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .









<http://eucommunity.eu/ontology>

  rdf:type owl:Ontology ;

  dcterms:created "08-04-2016"^^xsd:string ;

  dcterms:creator "Miltiadis KOKKONIDIS (miltiadis.kokkonidis@intrasoft-
intl.com)"^^xsd:string ;

  dcterms:description """The EU Community Policy Process Ontology captures
the EU Community project's conceptualisation of:

1) policy processes;

2) legislative procedures;

3) documents."""^^xsd:string ;

  dcterms:publisher "Intrasoft International S.A."^^xsd:string ;

  dcterms:title "EU Community Policy Process Ontology"^^xsd:string ;
```

```
  owl:versionInfo "1.0"^^xsd:string ;
.

<http://eucommunity.eu/ontology#PolicyModellingConcept>
  rdf:type owl:Class ;
  rdfs:label "Data type"^^xsd:string ;
  rdfs:subClassOf owl:Thing ;
.

<http://eucommunity.eu/ontology#UserSubmittedEntry>
  rdf:type owl:Class ;
  rdfs:label "User Submitted Entry"^^xsd:string ;
  rdfs:subClassOf owl:Thing ;
.

<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept>
  rdf:type owl:Class ;
  rdfs:label "Data type"^^xsd:string ;
  rdfs:subClassOf owl:Thing ;
.

<http://eucommunity.eu/ontology#Section>
  rdf:type owl:Class ;
  rdfs:label "Section"^^xsd:string ;
  rdfs:subClassOf <http://eucommunity.eu/ontology#PolicyModellingConcept>
;
.
```

```
<http://eucommunity.eu/ontology#Topic>

  rdf:type owl:Class ;

  rdfs:label "Topic"^^xsd:string ;

  rdfs:subClassOf <http://eucommunity.eu/ontology#PolicyModellingConcept>
;

.



<http://eucommunity.eu/ontology#PolicyProcess>

  rdf:type owl:Class ;

  rdfs:label "Policy Process"^^xsd:string ;

  rdfs:subClassOf
<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept>,
<http://eucommunity.eu/ontology#UserSubmittedEntry> ;

.



<http://eucommunity.eu/ontology#LegislativeProcedure>

  rdf:type owl:Class ;

  rdfs:label "Legislative Procedure"^^xsd:string ;

  rdfs:subClassOf
<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept> ;

.



<http://eucommunity.eu/ontology#Document>

  rdf:type owl:Class ;

  rdfs:label "Document"^^xsd:string ;

  rdfs:subClassOf
<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept>,
<http://eucommunity.eu/ontology#UserSubmittedEntry> ;

.
```

```
<http://eucommunity.eu/ontology#PolicyProcessStage>

  rdf:type owl:Class ;

  rdfs:label "Policy Process Stage"^^xsd:string ;

  rdfs:subClassOf
<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept> ;

.









<http://eucommunity.eu/ontology#id>

  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;

  rdfs:comment "A numeric ID that uniquely identifies instances of some
PolicyModellingConcept (not a globally unique identifier) "^^xsd:string ;

  rdfs:domain <http://eucommunity.eu/ontology#PolicyModellingConcept> ;

  rdfs:label "ID"^^xsd:string ;

  rdfs:range xsd:integer ;

.


<http://eucommunity.eu/ontology#title>

  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;

  rdfs:domain <http://eucommunity.eu/ontology#PolicyModellingConcept> ;

  rdfs:label "has data format"^^xsd:string ;

  rdfs:range xsd:string ;

.


<http://eucommunity.eu/ontology#description>

  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
```

```
  rdfs:domain <http://eucommunity.eu/ontology#PolicyModellingConcept> ;

  rdfs:label "has data format"^^xsd:string ;

  rdfs:range xsd:string ;

.





<http://eucommunity.eu/ontology#startDate>

  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;

  rdfs:domain
<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept> ;

  rdfs:label "Start Date"^^xsd:string ;

  rdfs:range xsd:date ;

.


<http://eucommunity.eu/ontology#endDate>

  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;

  rdfs:domain
<http://eucommunity.eu/ontology#ChronologicalPolicyModellingConcept> ;

  rdfs:range xsd:date ;

.




<http://eucommunity.eu/ontology#submittedDate>

  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;

  rdfs:domain <http://eucommunity.eu/ontology#UserSubmittedEntry> ;

  rdfs:range xsd:date ;

.
```

```
<http://eucommunity.eu/ontology#createdDate>
  rdf:type owl:DatatypeProperty, owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#Document> ;
  rdfs:range xsd:date ;
.




<http://eucommunity.eu/ontology#polarity>
  rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#Document> ;
  rdfs:range xsd:integer ;
.

<http://eucommunity.eu/ontology#polarityConfidence>
  rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#Document> ;
  rdfs:range xsd:double ;
.

<http://eucommunity.eu/ontology#subjectivity>
  rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#Document> ;
  rdfs:range xsd:integer ;
.
```

```
<http://eucommunity.eu/ontology#subjectivityConfidence>
  rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#Document> ;
  rdfs:range xsd:double ;
.




<http://eucommunity.eu/ontology#isCataloguedUnder>
  rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#PolicyProcess> ;
  rdfs:label "is in section"^^xsd:string ;
  rdfs:range <http://eucommunity.eu/ontology#Section> ;
.

<http://eucommunity.eu/ontology#isInSection>
  rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#Topic> ;
  rdfs:label "is in section"^^xsd:string ;
  rdfs:range <http://eucommunity.eu/ontology#Section> ;
.

<http://eucommunity.eu/ontology#hasLegislativeProcedure>
  rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
  rdfs:domain <http://eucommunity.eu/ontology#PolicyProcess> ;
  rdfs:label "has legislative procedure"^^xsd:string ;
  rdfs:range <http://eucommunity.eu/ontology#LegislativeProcedure> ;
```

```
.


<http://eucommunity.eu/ontology#belongsToPolicyProcess>
  rdf:type owl:ObjectProperty;
  rdfs:domain <http://eucommunity.eu/ontology#Document> ;
  rdfs:label "belongs to policy process"^^xsd:string ;
  rdfs:range <http://eucommunity.eu/ontology#PolicyProcess> ;
.
```